User's Guide™

THE MAGAZINE FOR CP/M & MS-DOS COMPUTER USERS

No. 15

$4.50

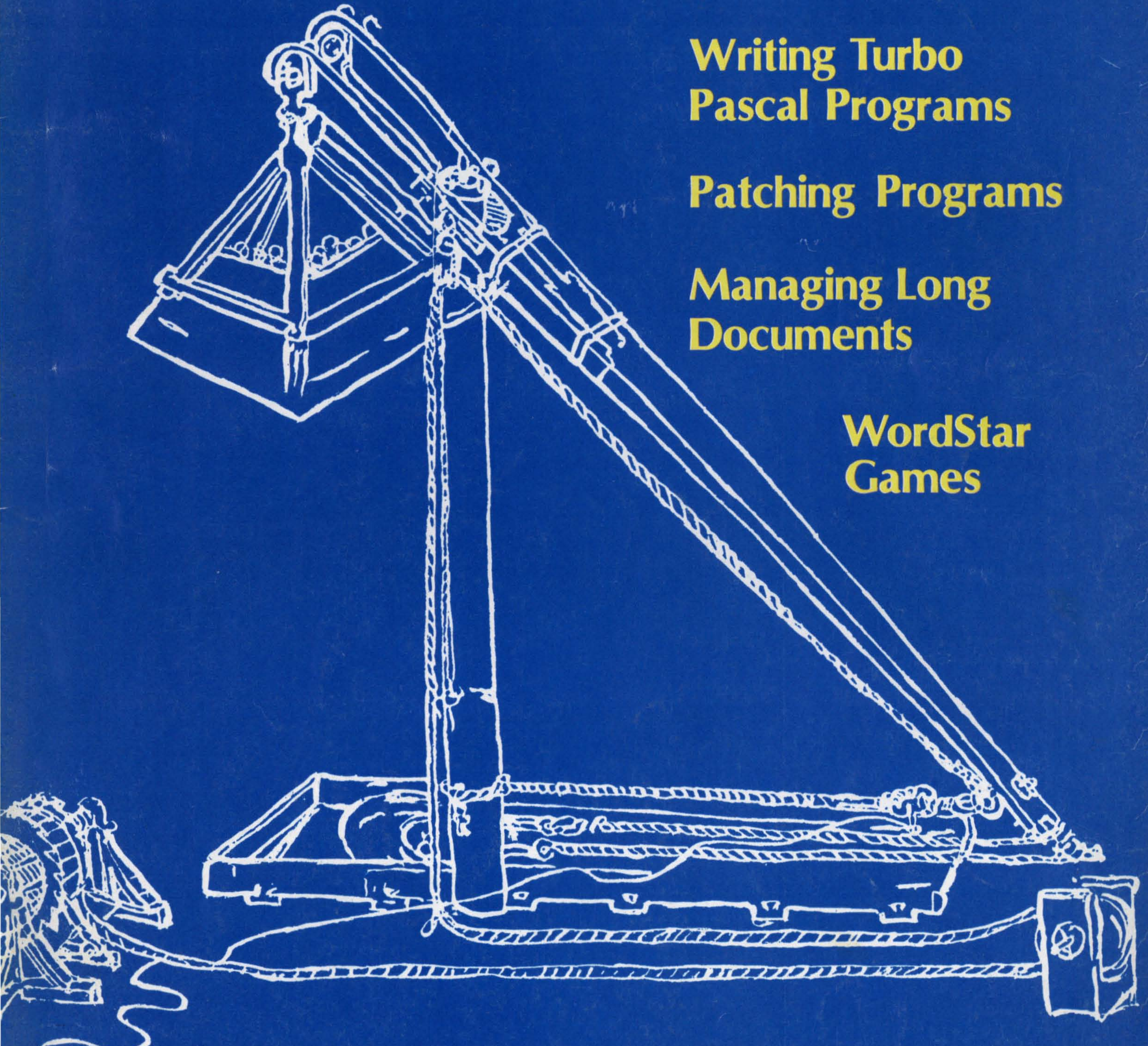
Z80 Assemblers Compared

Writing Turbo Pascal Programs

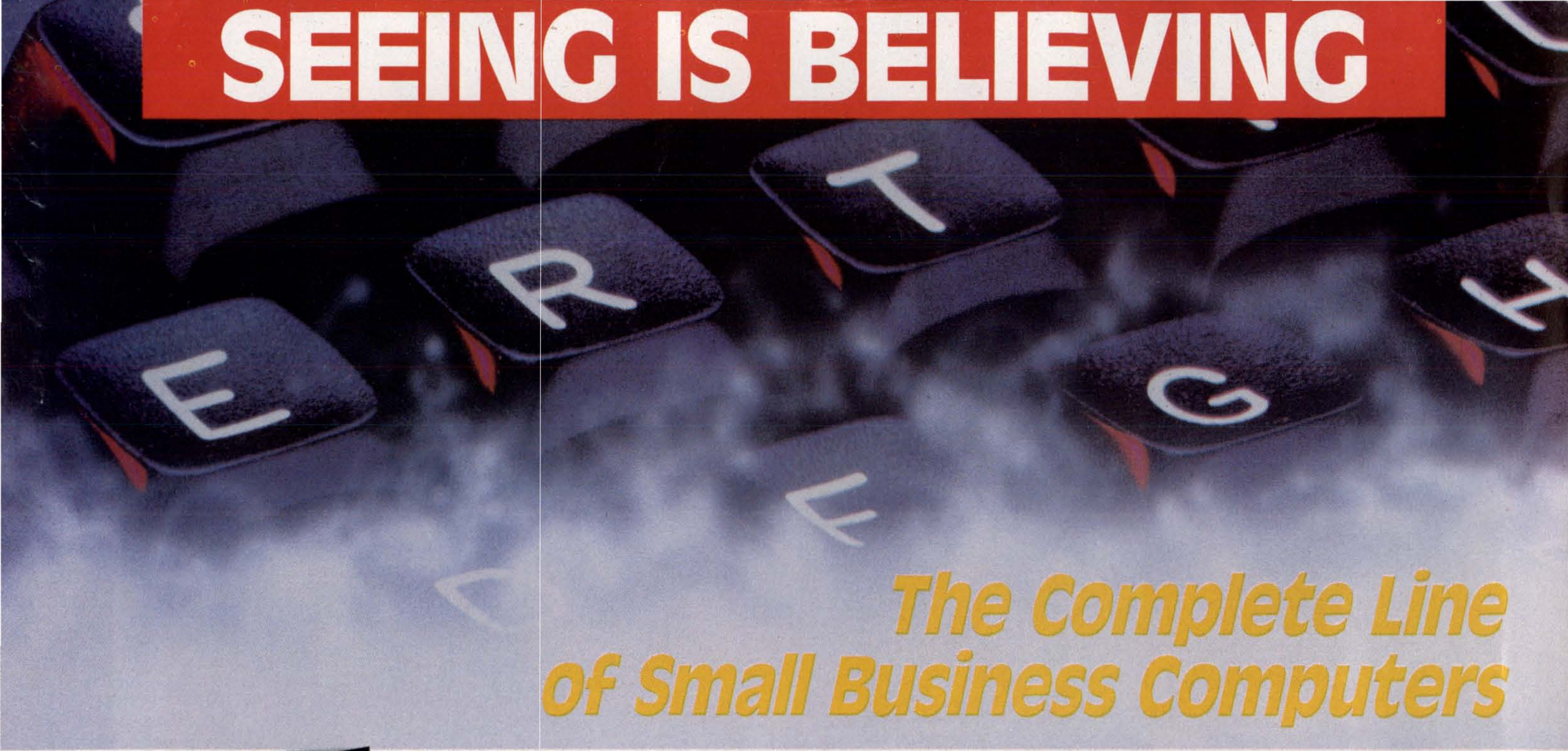
Patching Programs

Managing Long Documents

WordStar Games



"Study of Machinery" by Leonardo Da Vinci.



SEEING IS BELIEVING

The Complete Line of Small Business Computers

KAYPRO NEW 2

This small business computer provides: • A disk drive capable of storing 392K • A 9" green screen • Familiar typewriter-style keyboard with numeric pad • Connections for printer, modem, etc. • Software: WordStar,® M-Basic,™ CP/M™ 2.2, and Games • Optional Kaypro New 2 Upgrade Kit ($495) includes: A second disk drive and a complete software package.

COMPLETE $995

KAYPRO 2X

The Kaypro 2X, the professional business computer, comes with: • Dual disk drives providing 784K (400 pages) of storage • An easy-to-read, green screen • Familiar typewriter-style keyboard with numeric pad • A 300 baud modem • Real time clock / calendar • A complete bundle of business and personal software including: WordStar,® The Word Plus,™ DataStar,™ ReportStar,™ Mailmerge,® CalcStar,™ Microplan,™ M-Basic,™ C-Basic,™ S-Basic,™ CP/M™ 2.2, and Mite.™

COMPLETE $1595

KAYPRO 10

The Kaypro 10, the hard disk business computer, is complete with: • 10 megabyte hard disk capable of storing an astounding 5000 pages of data • Plus a 392K (floppy) for extra storage and hard disk back-up • A large, non-glare monitor • Familiar typewriter-style keyboard with numeric pad • Built-in modem • Real time clock / calendar • The same powerful package of ready-to-run software as the Kaypro 2X, plus dBASE II™ data base management already installed on the hard disk.

COMPLETE $2795

KAYPRO Robie

The Robie takes up less than half the space of a standard desk-top computer, but it's worth its weight in performance: • Two disk drives providing 5.2 megabytes of storage • Non-glare monitor • Built-in modem • Typewriter-style keyboard • And the same powerful package of software that's included with the Kaypro 10.

COMPLETE $2295

KAYPRO
QUICK-CREDIT
KAYPRO CORPORATION

Ask about the **Kaypro** Quick Credit Plan. You can finance a **Kaypro** computer for as little as **$38 per month.**

KAYPRO®
CORPORATION

For your nearest KAYPRO dealer, call TOLL FREE 1-800-4KAYPRO • In California, 1-800-9KAYPRO

Trademarks: CP/M, C-Basic, Digital Research, Inc.; The Word Plus, Oasis Systems; Mite, Microft Labs; S-Basic, Kaypro Corporation; Microplan, Chang Labs, Inc.; M-Basic, Microsoft, Inc.; WordStar, Mailmerge, ReportStar, DataStar, CalcStar, MicroPro International; dBASE II, Ashton-Tate.

Circle number 6 on the reader service card.

THIS IS THE COMPUTER WE WERE GOING TO INTRODUCE BEFORE YOU-KNOW-WHAT HAPPENED. ONE YEAR LATER, IT'S STILL AHEAD OF ITS TIME.

INTRODUCING THE OSBORNE VIXEN.™

The company that introduced the first portable business computer is back.

We've learned that building a $100 million corporation in a year can have its risks.

We've learned that this market doesn't need another IBM® clone.

And with all today's talk about windowing, 16- and 32-bit systems, we've noticed that the all time best-selling microcomputers are still the 8-bit Apple®s. They invite you to compare. So do we.

Like Apple, the Osborne computing concept has served certain people very, very well.

So well that over 100,000 of you have bought Osborne computers.

So well that to date, only a few computers have a bigger installed base.

The demand continues.

This is Vixen. The essence of the Osborne concept. Yet improved in every respect.

We took the best features of existing Osbornes and put them in a smaller case. With a bigger screen. 80-column display. And even faster response time. Result: The most powerful portable in its class. The Vixen is a fraction of the size of other Osbornes—yet it's *packed* with features lap-sized computers make you sacrifice.

Designed to endure.

Look what Osborne gives you now:

- Two double-sided, double-density, half-height *400K* disk drives let you store up to *100* typed pages per disk.
- 7-inch diagonal, power-saving built-in amber monitor.
- 80-column, 24-line display with easy-to-view 8 x 10 dot characters.
- Serial *and* parallel ports let you connect the Vixen to any printer or modem.
- CP/M® operating system, for access to the huge library of CP/M productivity software.
- 90-day warranty.

All in one neat little package that is smaller than an IBM Selectric typewriter—and about half the weight!

And all for $1298.

Take it away.

The Vixen is data-disk compatible with IBM PCs—and more.

Free Media Master™ software lets you exchange data diskettes between the Vixen and over 200 other computers, including the IBM PC.

Your computing capabilities just expanded 200 times over.

Free productivity software: An Osborne tradition.

When you buy the Vixen, we give you software tools for writing, for creating spreadsheets, custom programming, graphics and plain having fun.

You get WordStar®/MailMerge,® SuperCalc 2,™ MBasic,™ Desolation, Media Master,™ Osboard™ and TurnKey. Free.

Free Membership in FOG, the First Osborne Group: Support network, newsletter, and access to a huge library of public domain software.

Osborne will buy every Vixen owner an annual FOG membership, making you part of the largest user group around.

Telephone experts for answers to your computing questions. You can attend group meetings at over 300 locations. And take advantage of thousands of public domain software packages in FOG's ever-expanding library.

Fellow FOG members are a terrific source for tips on getting more from your Osborne. Thinking of buying a new program? Check its track record with FOG members who already use the package before you invest.

Here it is: Portable, powerful, reliable and data-compatible.

If you liked Osborne computers before, you will absolutely, positively, ***love*** *this new machine.*

OSBORNE

YES, I want to know more about the Vixen, Osborne's newest computer.

Osborne Computer
Corporation
42680 Christy Street
Fremont, CA 94538

Name ______________________
Address ______________________
City ______________ State ______________ Zip ______________
Phone number (   ) ______________________

☐ ***Send me more information on the Osborne external hard disk.***

Trademarks: Vixen—Osborne Computer Corporation; Media Master—MDG & Associates; SuperCalc 2—Sorcim/IUS Micro Software; MBasic—Microsoft Corporation; Osboard—DG Systems.
Registered Trademarks: IBM—International Business Machines Corporation; Apple—Apple Computer, Inc.; CP/M—Digital Research, Inc. WordStar/MailMerge—MicroPro International Corporation.

Circle number 11 on the reader service card.

User's Guide

Number 15 (Volume 3.3) **June 1985**

Tony Bove & Cheryl Rhodes
PUBLISHERS & EDITORS

Steve Rosenthal, Kelly Smith,
CONTRIBUTING EDITORS

John Barry, Dana Blankenhorn,
Benjamin Cohen, Thomas Howe,
Bruce Morgen, Arthur Naiman,
Jonathan Sachs, Bob Wolff
WRITERS

Sahnta Pannutti
DESIGNER

Lon Shoemaker (cartoons),
Paul Winternitz (photos)
ART & PRODUCTION

Annie Warren
ADVERTISING SALES

William Shoemaker, Kathy Slusser
CIRCULATION

Julie Swanson
ADMINISTRATION

Hugh Daniel, Dorothy Hays
RESEARCH

Dan Shoemaker
OFFICE

User's Guide , ISSN 0747-8534, is published six times a year by TUG Inc., Box 5245, Redwood City, CA 94063, (415) 364-0108. Second-class postage paid at Redwood City, CA 94063 (USPS 689-550).
Entire contents Copyright © 1985 by TUG, Inc. All rights reserved. Reproduction of material appearing in *User's Guide* is forbidden without explicit permission. Send all reprint requests to Reprints, User's Guide, Box 5245, Redwood City, CA 94063.
SUBSCRIPTION INFORMATION:
Single copy price $4.50 ($6.00 foreign or U.S. first-class). Subscription for six issues (one year) is:
U.S. — $21 if payment accompanies order, $25 if subscription is billed (second class). Add $10 for first-class delivery.
CANADA & MEXICO — $27 in U.S. currency drawn on a U.S. bank, or international postal money order; payment must accompany order (second class). $35 for first-class delivery.
FOREIGN — $40 for air mail subscription. Payment (U.S. currency drawn on a U.S. bank, or international postal money order) must accompany foreign orders.

Inquiries about bulk, retail sale, and group rates are encouraged. Address inquiries to Circulation Dept., **User's Guide, Box 5245, Redwood City, CA 94063.** Allow 6-8 weeks for processing of order.

POSTMASTER: Please send address changes (Form 3579) to User's Guide, Box 5245, Redwood City, CA 94063.
CP/M and CP/M-86 are registered trademarks of Digital Research, Inc. IBM is a registered trademark of International Business Machines. MS-DOS is a trademark of Microsoft. WordStar and MailMerge are trademarks of MicroPro, Intl. Turbo Pascal is a trademark of Borland International.
User's Guide and Impatient User's Guide are trademarks of TUG Inc.
User's Guide (TUG Inc.) is not affiliated with Digital Research, Microsoft, or any hardware or software manufacturer or distributor.

CONTENTS...

EDITORIAL

Panacea For "Power Users" 10

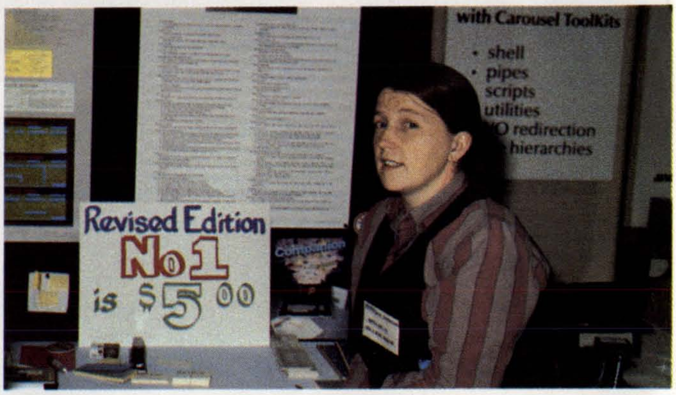
LETTERS 14

FOR STARTERS

Customizing & Patching 11

SLEAZE PATROL

Auction Blues 12
by Tony Bove

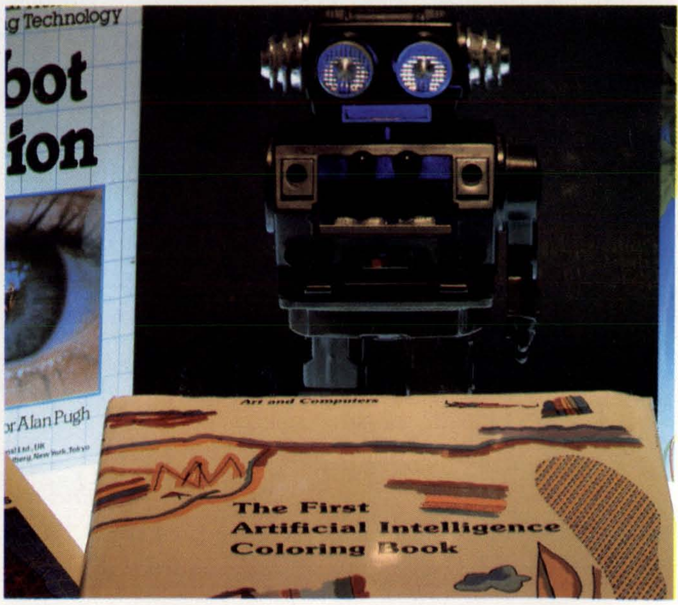


ESCapades

West Coast Computer Faire and SoftCon 4

BOOKS

AI Books Thrive in PA 73
by John Barry



Issue 15 (Vol. 3.3)

TUTORIAL

WordStar Games: Part 3 **24**

by Jonathan Sachs

Word processing techniques for handling large WordStar documents.

eCode = '{mc}'
'{ql}'; {*
dLeft = '{cf3}';

TUTORIAL

Turbo Pascal: Part 2
Text Conversion **58**

by Tony Bove & Cheryl Rhodes

This installment describes programming techniques and introduces a Pascal program that converts text printing codes into typesetting codes.

TUTORIAL

Patch It Up **16**

by Richard Drakeford

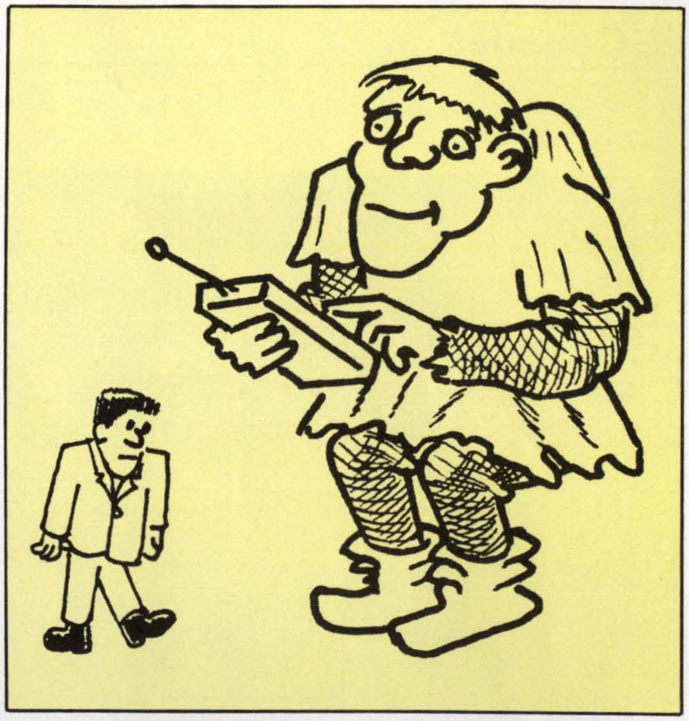
Have you ever wondered what "patching" means, and what it can do for you? Patching is explained, with common WordStar patches.

REVIEW

Z80 Assemblers
For 8-bit CP/M **32**

by Edward Joyce

A roundup of CP/M-80 assemblers: heavy duty programming tools for writing and debugging Z80 assembly language programs.



ADVANCED USER

The Advanced User
In the 8-bit World **66**

by Bruce Morgen

The wonders of ZCPR (a system enhancer) and LRUN (running programs in libraries).

ANNOUNCEMENTS

New Products **54**

MICRO USER'S MARKET **76**

ADVERTISER INDEX **76**

Typefaces

We use `this Typeface` to show commands and expressions you type at your keyboard, and `this typeface` to show displayed messages.

In summaries we use *italics* along with `this Typeface` to show optional parts of commands and expressions. Keywords are usually in UPPER case letters, with your words and data in Upper/lower case.

We also use the symbol ↵ to represent the standard Return key (CR or ENTER on some keyboards), and the caret symbol ^ to represent the Control key (CNTL, CTRL or ALT on some keyboards).

Paul Winternitz

Through the golden arches of Moscone Center is revealed a thick crowd. The Faire was attended by over 50,000 people in four days — a new record.

West Coast Computer Faire and SoftCon

The computer shows are growing up fast, and some are falling flat on their faces. The Tenth West Coast Computer Faire in San Francisco was a success (record-high attendance) but was largely ignored by the press. The SoftCon show in Atlanta had a good conference section but near zero attendance, and it may not be around next year. The following are reports from SoftCon (Dana Blankenhorn) and the Faire (the editors with Jonathan Sachs).

Koppel Condemns Computer Con at Softcon

by Dana Blankenhorn

Softcon has burst onto Atlanta like the great April Fool in the sky. The floor was very empty. Apple was inviting software developers only to a skybox with a Mac Office and Mac Officials in blue suits. The Mac Following also huddled at the Omni this morning, and new Mac Fellows were signing non-disclosure agreements to get in.

Softcon opened (and probably closed) with a panel discussion led by ABC's Ted Koppel, the Phil Donahue of the Midnight Hour. The loge section was packed, but curiously the balcony in the Sidney Marcus Auditorium was empty.

Present were Esther Dyson (influential newsletter editor), Fred Gibbons of Software Publishing, John Sculley of Apple, and the inimitable Mitch Kapor of Lotus Development Corp., who later regaled everyone with tales of managements buying copies of 1-2-3 by phone at $500 per. (Softcon had promised us Jack Tramiel, but the Fat Man failed to show up. Fans of the Fat Man, his cigar, and the New Atari Corp. were told he was off in Europe, maybe lining up some financing.)

Mr. Koppel kept trying to pin this esteemed panel down to the question of what they were going to do to get him to use a computer, but he found his quarry more elusive than Rev. Tutu and Mr. Botha put together. "I'm the last hold-out," he said. "I don't even use an electric typewriter. My manual is 50 years old — an old Smith-Corona.

"Your biggest task is convincing people like me you have something that's not only valuable, but that I can learn to use. It's estimated that there are 60 million functional illiterates in the United States today—not computer illiterates, but people who can't read or write enough to function in our society. Part of your problem is changing that, and bringing us into the second half of the 20th century."

Ted never did get a satisfactory answer to his question of "what do I need with this box," although he did keep trying.

Copyright © 1985 Dana Blankenhorn
Copyright © 1985 Bove & Rhodes

ide
WORDSTAR SUMMARY
MURPHY'S COMPUTER LAW
Do More
Learn the style of Unix with Carousel ToolKit
shell
pipes
scripts
utilities
I/O redirection
hierarchies
Revised Edition
No 1
is $5 00
User's Guide

User's Guide at the opening of the Tenth West Coast Computer Faire.

"You fellows did us in over the last five years," Koppel said. "You conned us into buying home computers which are now gathering dust in homes around America."

Unfortunately, our heroes on the panel could not convince Mr. Koppel to buy. Instead, they mostly congratulated themselves and said they'd just have to work on it.

John Sculley: "We have 10% computer literacy in this country, the highest in the world. We're where the auto industry was in 1915, before automatic transmissions." Mr. Sculley predicted that the computer will become a learning machine, helping us move among four or five careers in the course of a lifetime. (John should live so long.)

Mitch Kapor: "Who knew that in 1978 the electronic spreadsheet, invented by Mr. Daniel Bricklin down here in the front row (and a few inches below his foot) would become a $500 million business?"

Fred Gibbons: "Any time you see a need products will be created to fill it. We put a key piece of information on our electronic mail service, and made people go to the service to read it. That kick-started the process of using electronic mail. The consensus is the market will be in information."

Esther Dyson: "So far the Personal Computer is not a physical machine. It's an intellectual machine. It's automating things you might do by hand." Later she agreed with Mr. Gibbons. "It's not so much a question of applications as information. The guy who bought the first telephone had a problem, too."

Mr. Koppel knew his question still had not been answered. "I don't want to flog a dead modem," he said, "but what you seem to be saying is the 'home computer market' as such is dying on the vine." The only "home uses" of a computer involve bringing the office home, in other words.

And that's where they left it. Pity the Fat Man missed it. Wish I had, too.

As the session wound down to some boring Q&A I wandered down to the press room, and lo and behold it was like I died and went to heaven. A half dozen waiters were filling dozens of glasses with red, white, and rose wine, while in front of them stretched a 12-foot long fruit, cheese and crackers spread. Minutes later the panelists and Mr. Koppel were walking into the room.

Made brave by wine, I told Mr. Koppel when he could expect to see a personal computer on his desk. "When your boss says he can no longer afford to have someone else punch buttons for you. Or when he says you no longer have 30 minutes to write your opening tonight, but 15 minutes. In short, when Capital Cities starts tightening belts—they're very good at that, you know." I think that last hit home, because Mr. Koppel turned a few interesting shades of red, yellow, and green, and his right eyebrow arched a little, like Mr. Spock's on Star Trek.

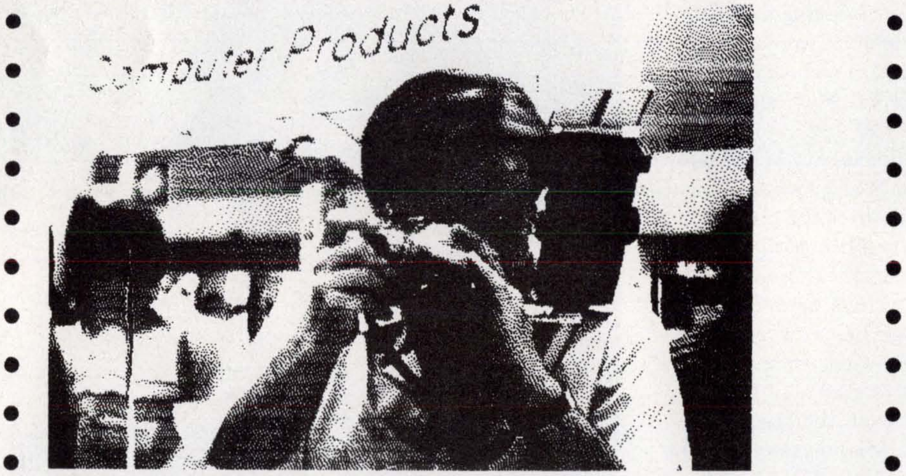
Paul Winternitz

The User's Guide booth enjoys brisk sales.

The neatest booth on the floor was Odesta's, where some Mac Software was actually up on desks, running. Here and there a few other believers were passing Mac Documents back and forth. But if you treat these distractions as you would the Hare Krishnas at the airport you can pass on by and discover there's really not a whole lot of new things here. There's talk of

Paul Winternitz

Paul Winternitz shoots his digitized image as shown on the Apple Mac.



The digitized image printed on an Apple Imagewriter printer.

Expert Systems, and networking, and there were some mini and mainframe software programs shown, but I couldn't understand a jot about them.

The whole Softcon floor took up less than half the World Congress Center's west showroom—maybe there were 250,000 square feet there, but the hallways were *huge*, the booths spread out with extra elbow room, and the empty concession stands seemed to stretch out forever.

On the gew gaw beat, Software Response, which mails sets of advertising post-cards around the country, gave me a package of Life Savers. Gold Hill Computers of Cambridge, Mass., makers of Golden Common Lisp, an AI language, gave me an artsy poster I may put on my wall. Something done by Karen A. Prendergast in 1981, American modern nothingness. Quadram gave me a soft plastic ruler, clear and with blue and yellow in it, showing both centimeters and inches. Somebody gave me a little plastic football, but I gave it to one of the little kids down the street, who tried to play stickball with it.

That's about it, except for the CP/M exhibitors. There were 15 exhibitors listed in the program as showing CP/M compatible software. Here they are:

CPAids Inc. (1061 Fraternity Circle, Kent, OH, 44240 (216)678-9015) showing accounting software for CPAs, a "total network" they say. *(Note—reviewed in issue #14).*

Construction Data Control (6140 Northbelt Parkway Suite A., Norcross, GA 30071 (404)448-4722) showing estimating and accounting software for construction firms.

Dagar/Software Development Corp. (270 Amith Road, Woodbridge, CT 06525, (203)397-0500) showing retail pharmacy packages.

Digisoft Computers Inc. (1501 3rd Avenue, New York, NY 10028 (212)734-3875) showing specialized electronic mail software.

Innes Systems (21 East 40 St., New York, NY 10016 (212)679-6180) a vertical market software and hardware distributor.

Liberator Software (537 Castro St., San Francisco, CA 94114 (415)864-1942) showing an integrated retail sales, inventory control and management system.

Limicon Inc. (144 Hampton Avenue, Toronto, Ontario M4K 2Z1 (416)481-7859) showing a graphics packaged called ProDraw, last seen next to words like NAPLPS and videotex.

MC Software (P.O. Box 1377, Davis, CA 95617 (916)756-8104) showing manufacturing and account software under Concurrent DOS.

Micro Manufacturing Systems (2600 Corporate Exchange Drive, Suite 190, Columbus, OH 43229 (614)895-0738) showing an integrated manufacturing software system.

Northouse Industries Inc. (7110 W. Fond du Lac Avenue, Milwaukee, WI 53218 (414)438-0610) showing software for farmers with names like HerdStar, CropStar, BeefStar and the like.

OA Software (2185 The Alameda, San Jose, CA (408)554-9086) showing software for lawyers.

Omega Information Systems Inc. (P.O. Box 450125, Atlanta, GA 30345 (404)938-4645) showing ADAM and ADAM jr., software for churches and synagogues.

The Programmer's Shop (128 Rockland St., Hanover, MA 02339 (617)826-7531) selling books and compilers for C, Ada, Modula 2, Prolog, etc.

Rational Systems Inc. P.O. Box 480, Natick, MA 01760 (617)653-6006) showing a high-performance C called Instant-C. (I suppose the name XTC is already taken.)

Sawhney Software (888 7th Avenue, New York, NY 10106 (212)541-8020) selling professional-quality tax planning and

Continued on p. 8

WHAT THE NEW YORK TIMES SAYS ABOUT

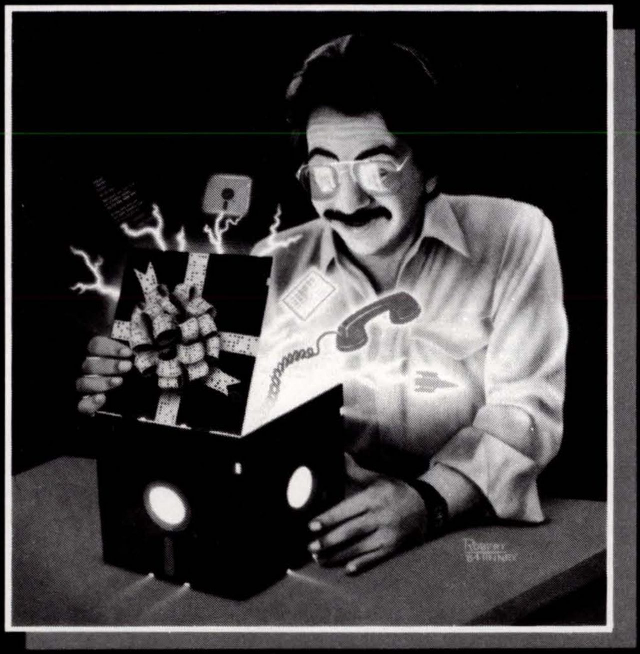
FREE SOFTWARE

"Software does not have to be expensive. In many cases, a program just as useful as the one that costs $495 at the local computer shop may be available for the price of a single floppy disk through public domain libraries.

"But how does one go about finding this treasure of free software, including word processors, spelling checkers, data bases, accounting packages, spreadsheets and above all games, games, games? A smart place to start is with FREE SOFTWARE by Tony Bove, Cheryl Rhodes and Kelly Smith.

"The authors give clear advice on how to get the software and how to use it. They also perform a valuable service by sifting through mounds of junk programs that clog the public domain, saving the user time as well as money."

★ AN ALTERNATE SELECTION OF THE MACMILLAN SMALL COMPUTER BOOK CLUB ★



Available now in stores near you!
55948-6 • 320 pp. • $9.95

Distributed by Simon & Schuster Mass Merchandise Sales Co.
1230 Avenue of the Americas • New York, N.Y. 10020

TONY BOVE, CHERYL RHODES, AND KELLY SMITH

Circle number 1 on the reader service card.

Paul Winternitz

Jim Warren, Gary Kildall, Adam Osborne, Lee Felsenstein and Gordon Eubanks led a lively panel on the history of personal computing.

Continued from p. 6

personal finance software.

Who says CP/M is dead?

F.B.I. and No Responsibility at the Tenth Faire

by Tony Bove & Cheryl Rhodes
with Jonathan Sachs
Photos by Paul Winternitz

The West Coast Computer Faire has a tradition to uphold. Seven years ago the show was for hackers and industry people, with little or no resources for people interested in computing but unwilling to program in assembly language or fool with front panel switches.

This year's Faire shows how the industry has changed dramatically. Gone for the most part were the hacker-dominated small businesses, and "non-techies" outnumbered techies ten to one. The industry has matured, some say prematurely.

The Tenth West Coast Computer Faire was held in the Moscone Convention Center in San Francisco, which is a huge underground pit. The latest in technology was featured in large display areas centrally located; they were surrounded by aisles of small booths, many of them holding bargain basement sales.

The highlights of this show are usually some of its conference sessions. This year we saw the lively "retrospective" panel trade jibes with each other and give a humorous perspective on the last ten years of microcomputing. This panel included Jim Warren (Faire founder), Gary Kildall (Digital Research founder, creator of CP/M), Adam Osborne (who else?), Lee Felsenstein (designer of the Processor Tech Sol, the Osborne 1, and spokesperson of the Community Memory Project), and Gordon Eubanks (creator of CBASIC).

Other sessions were little more than publicity spots for companies and their products, but the large sessions (Jerry Pournelle, the retrospective, and the panel on "Computers: Panacea or Poison") were packed and well received.

The show was very crowded for most of the four days, with a record attendance of around 50,000. The Faire started on a weekend, drawing record crowds of the "general public," and extended through Monday and Tuesday, drawing smaller crowds of "industry types" and business people.

It had the feel of a county fair, but it sparkled with chic computer gear, including a lot of PC and Mac accessories like digitizers, graphics devices, the new laser printers and hard disk drives. Exhibitors included

The F.B.I. booth, not listed in the program, occupied space down the aisle from the deserted Computer Professionals For Social Responsibility booth.

FEDERAL BUREAU OF INVESTIGATION

large retail outfits selling lots of cheap printer stands, disk holders and other computer paraphernalia along with inexpensive software and games.

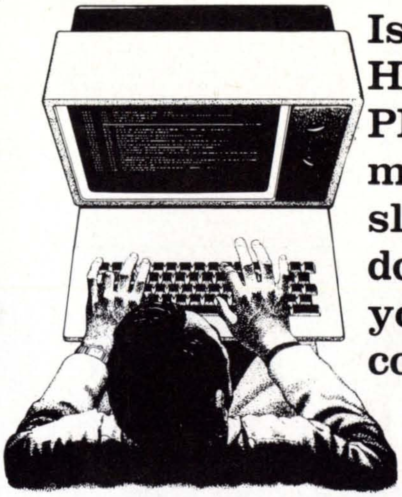
The prices were excellent: we saw a five-megabyte hard disk drive (*sans* power supply, controller or case) for $99; an excellent $35 program for merging mailing lists with WordStar form letters called PUNT (written in MailMerge codes!); large quantities of public domain software at $5 a disk.

There weren't a lot of new products, and only a few of the big computer manufacturers were there, including Apple, IBM, AT&T, NEC, Commodore, Altos and Osborne.

Although Atari wasn't exhibiting, an Atari user group mini-booth was very crowded because it had a prototype "Jackintosh" computer running GEM (Graphics Environment Manager from Digital Research). This promises to be a Mac killer for about $600.

The least crowded was a deserted booth for "Harmless Software"—was it mute witness to some corporate tragedy, a social comment ("the only good program is a dead program..."), or an April Fool's joke?

This wasn't the only deserted booth—the Computer Professionals For Social Responsibility was also empty. Right down the aisle was the Federal Bureau of Investigation, whose booth personnel's badges had no names, just "FBI". Where they looking for suspicious criminal-hackers, trying to sniff-out the "entrepreneurs" who specialize in exports to Bulgaria, or just making an effort at public relations? "They have a sense of humor," was all that Jerry Pournelle would say. Perhaps they were recruiting...



Is the HUNT & PECK method slowing down your computing?

Master typing with . . .
TYPE-RITER the CP/M typing instruction and drill program

Menu Driven • Random Access Generation • 64,000 Lessons Scored Escape • Accurate Timing & Scoring • Records all Timed Test Scores • For Individual or Classroom Use

ORDER NOW — Call or Write

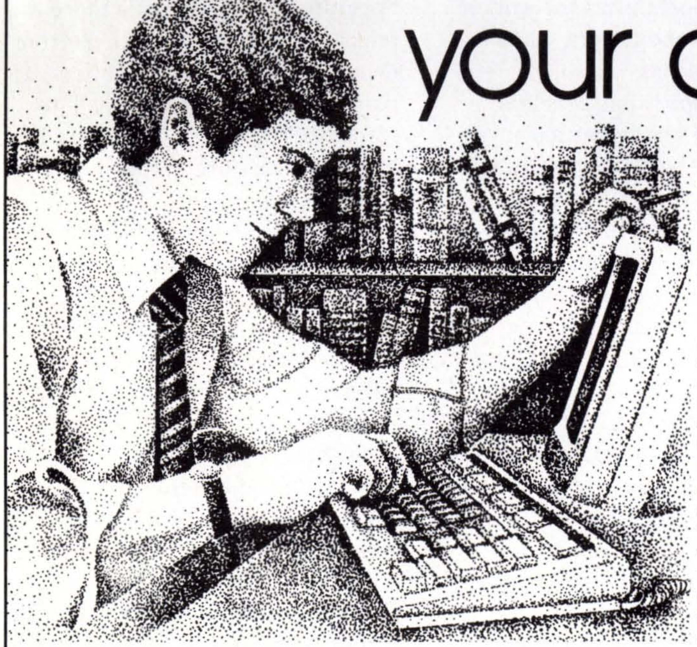
4418 Chapman, Suite 311
Orange, CA 92669
(714) 639-8106

LIST PRICE $29.95

CP/M is a registered trademark of Digital Research, Inc.

Circle number 10 on the reader service card.

Reach for the height of your craft...



As an idea craftsman, you use your mind like a skilled pair of hands. You take hold of concepts, then manipulate them to discover, refine, convey dynamic new thoughts.

Now, the tool that can strengthen your creative grasp is well within your reach.

KAMAS™ a revolutionary outline processor from KAMASOFT*, supports your thinking process and keeps you in touch with your ideas. That's because KAMAS is designed to work the way your mind works—naturally.

Begin by brainstorming. KAMAS enables you to jot down ideas quickly, as you think of them. If you want to elaborate, you can add text with full screen editing

Then develop your ideas using a familiar outline format. Change the structure as easily as you change your mind. Move an idea and all attached text moves with it.

KAMAS puts full control at your fingertips. You can keep track of your main line of thought by collapsing the details from view. Then, expand the outline to develop the specifics when you need to. By collapsing and expanding portions of the outline, you can maintain an overview and literally see how your ideas fit together.

And KAMAS is fast. Your ideas remain at your fingertips. KAMAS can locate and retrieve by keyword—even a misspelled one—in less than a second per topic file.

Use as much of the power as you feel comfortable with. KAMAS is menu-driven with over 100 on-line help screens. But you can also open the hood to find a high performance programming environment with the additional horsepower you may need to get the job done. An active user community and the KAMAS Report newsletter keep you in touch with the latest KAMAS applications.

Achieve a commanding vantage point...

KAMAS, Z80, & CP/M are trademarks of KAMASOFT*, Zilog, & Digital Research respectively.
* formerly Compusophic Systems.

Get your hands on KAMAS now. Send $147 plus $4 for S/H. Or call (503) 649-3765 for VISA or Master-Card orders. KAMAS is available for many Z80, CP/M computers. Ask about your system.

KAMASOFT*
2525 SW 224th Ave., Dept. 131
Aloha, OR 97006

...with KAMAS™

Circle number 3 on the reader service card.

EDITORIAL

Panacea For "Power Users"

The personal computer was designed by people who had had it up to here with the IBM idea — the monolithic control of information.
— Stewart Brand

The new criminals will be those who know how to fiddle with computers — those will be the dangerous people in society.
— George Morrow

I think they're great and a lot of fun and a lot of people get to make a lot of money who would not make money otherwise.
— John Dvorak

All the problems associated with computers seem to be actually problems in the way this society uses them.
— member of the audience

These panelists, speaking on "Computers: Panacea or Poison?" at the West Coast Computer Faire, are (like us) involved in the computer industry. One could argue for or against their supposed neutrality on the issue of whether computers are "good" or "bad" for human beings. But their perspectives summed up the popular themes in this on-going debate.

You can believe that computers are to blame for the illiteracy, worker alienation, industrial pollution and unemployment. Or you can believe that computers are wonderful tools, innovative and inspiring, able to liberate office workers from boring, tedious work.

You might believe *both* are true. Computers are blamed for many things, but nearly every atrocity can be shown to be an example of the new technology in the hands of narrow-minded people, implemented in old-fashioned and inappropriate ways.

In some offices where personal computers are used every day, the operating procedures are rigidly defined, and the workers use the computers in predictable, accountable ways. Perhaps their keystroke rates are monitored, or their output scrutinized for errors, or the typists have to relate to mnemonic codes that bear little or no resemblance to the data they represent. These are examples of computers reducing the opportunities for worker individuality and judgement when it comes to applications.

We always hoped that personal computers would change office environments for the better by putting workers in control of their work, allowing them to design their applications. This is happening in some corporations but mostly in small businesses and professions.

But today's IBM-dominated "mature" industry is more interested in making office workstations that work the same way as other workstations. The computer manufacturers are promoting standard operating procedures and rigid menu-structured applications, all in the name of "user friendliness."

The first users of personal computers believed that they were liberating themselves from the tyranny of the monolithic control of information. They cared about "user friendliness" in the context that the machines should be flexible and full-featured as well as easy to use — with function keys that could be programmed, user environments that could be customized, etc.

Today there are personal computer "operators" who do little more than select pre-defined operations from menus, type memos and letters, and plug numbers and formulas into spreadsheets. These people are hardly liberated from drudge work — they do it every day.

There are also those people who are called "power users" by the label-happy marketing researchers. These are the users who customize their work environments and more interested in functionality and customization.

If office workers can learn more about personal computers than merely how to use their menus, they can change things they don't like and make their work more interesting. In the process they can learn a lot more about the business they conduct, largely because they are empowered to make decisions and solve problems. This is how liberation can take place.

Customizing & Patching

There must be many of you "out there in beginner's land" who feel as one reader does, that "I know how to do the simple things with computers, but there's this wall of obscurity between me and the technical information on how to improve the way I do things." This wall separates those who type data into computers from those who can customize their computer operations.

Where do you start? How can you get the experience you need to vault over this wall without hurting yourself? We suggest that you learn about customizing the programs you use, and perhaps tackle a data conversion problem to gain experience with a programming language. This issue might be a painless way to begin learning about setting up applications.

In this issue we present a tutorial on "patching" using the ultimate CP/M customizing tool: DDT (yes, the pun is intended to be "bug-killer"). DDT is supplied with CP/M (one of the advantages a CP/M system enjoys over big brother MS-DOS is this handy utility) and lets you examine and change memory locations.

This means you can bring a program into memory and *change* it to suit your application. You can then use CP/M's built-in SAVE command to save the modified version of the program. This is called "patching". You can patch programs like WordStar to get at the custom features available in WordStar.

For a more detailed look at patching with DDT and SAVE, see issue #8. For more details on installing WordStar, see issue #2.

We also present part three of "WordStar Games" which gives instructions on how to deal with large documents — useful for anyone who wants to set up large-scale word processing operations (writers, lawyers, consultants, editors, small business managers, etc.). The previous "WordStar Games" installments appeared in issues #11/12 and #13; related articles on WordStar and MailMerge appear in issues #2, #3 and #6.

If you read part one of our Turbo Pascal tutorial (issue #10 on "Amateur Programming"), you will be happy to see part two in this issue. Turbo Pascal is possibly the first powerful and affordable tool for anyone to learn programming with. You can write entire application programs in this language. This installment presents a word processing problem and its solution: how to convert a text file's printing codes into typesetting codes (or laser printer codes).

You may not need to convert text files for typesetting or fancy printing, but you may need to take a text file or data file created by WordStar or dBASE II and somehow change it to make it compatible with another word processor or data base management system. You are faced with a customization task because you can't rely on a general purpose utility to perform such a conversion. Sometimes the best way to process large chunks of data is with a "macro" operations language (like dBASE II) or a real application programming language (like Turbo Pascal).

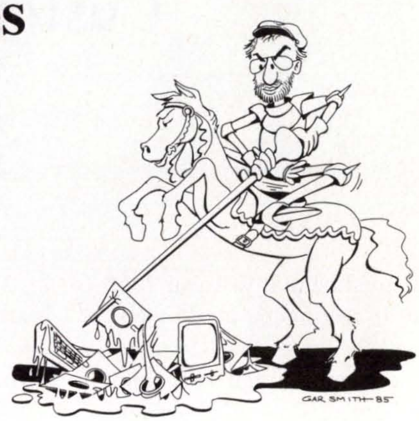
Ed Joyce reviews eight CP/M-80 assemblers for those readers who want to discover the power of assembly language. This is perhaps the most advanced article because it deals with assembly language tools; however, you don't have to know how to program in the language to learn about the benefits of one utility over another.

The new "Advanced User" column by Bruce Morgen rounds out our coverage of customizing system operations. He covers the ultimate CP/M customization: the complete replacement known as ZCPR3. This column will continue to provide information for people who want to improve the performance and increase the functionality of their systems.

Finally, John Barry reviews a plethora of books on artificial intelligence (keeping with our theme of improving our systems — thanks John). The basic ingredient for intelligence is the *ability to learn from experience.* In the Turbo Pascal tutorial we present a crude example of a program that modifies its execution according to its past experience. These books point the way toward a fuller understanding of the concepts of artificial intelligence.

SLEAZE PATROL/Arthur Naiman

Auction Blues



Arthur Naiman is on a trip through Central America for a few weeks and has graciously consented to having Tony Bove write a sleazey item as a "guest contributor." Let us know if you think he still needs to take Arthur's course in "effective sneering." Arthur has made it back in one piece and should have another real sleazey item for the next issue.

I went to an auction in Silicon Valley the other day, and learned first-hand how it feels to be ripped off and how to avoid it next time.

The auction was held by the Ross-Dove company in a huge customs warehouse in San Jose, California. A firm that sold computerized medical equipment and video imaging systems had somehow lost its footing in the Valley and was now reduced to auctioning off its lab equipment, office furniture and computer systems. As I walked into the cavernous darkness out of the blazing Valley sunlight I felt the cold shiver of bankruptcy.

This auction felt more like a wake. Cork boards still had calendars pinned to them. Desks still had pens and writing pads. Appointment schedules were frozen in time. Rolodexes stood empty. All of the black office telephones were put in one box. All of the white drawing boards were leaning up against a small forest of free-standing wood and metal bookshelves.

Brand new TeleVideo 910 terminals were still packed in their boxes. Dusty ADDS Viewpoint terminals, one with a partially scraped-off Apple logo on the keyboard, stood silently in a row, giving off the aura that they'd paid their dues and performed well, and that now they should be somehow "set free."

I had come for the terminals, but I also felt attracted to other things. This is the danger of going to auctions: you are suddenly captivated by the thought that all this stuff was selling for at least one-half, possibly one-fourth or even one-tenth of what they would cost in a store. Consequently you want to buy as much as you can afford.

I wandered through the maze of goods, looking over boxes of miscellaneous cables, electronic parts, lab tools, machine shop benches, formica-top office tables, counters, stools, drafting boards, heavy x-ray scanning equipment... I wanted one of the portable nine-inch Panasonic monitors for my Osborne 1. The two formica-topped tables in the back looked good. How about that funky light table?

It was early morning, and the list of items (auctioneers number each "lot" or item to sell and provide a list) was very long. They always start at the top of the list and it goes all day. It's usually crowded in the morning—nearly a hundred people looking over the merchandise. Lots of office furniture and high-ticket drafting boards are first on the list and the bidding is very high. My terminals, the funky light box, the Panasonic monitor and the tables were all at the end of the list, so I took a long lunch.

Don't buy anything at an auction that can be opened up and stolen from, unless you can cart it out immediately.

I knew I wasn't missing anything. Many people leave after the first hour or two, and never come back. I enjoyed a long lunch and came back refreshed. It was good that I'd done this—no one except Ross-Dove employees and warehouse employees were allowed to go to the bathrooms, which were in the roped-off sections of the customs warehouse.

When I got back the hundred had dwindled to about thirty hard-core male buyers of medical and computer equipment (it seems that only men are interested in outdated technological junk). All the medical and video imaging stuff had been auctioned off. I started the bidding on the Panasonic at $5, but I fought two others for it until the bidding went past $40 (one of them got it for $75, which is more than half the price of a new one). At first I felt like I'd wasted the day by not buying anything yet—this is precisely how the auctioneers want you to feel.

It was close to 4 p.m., the auctioneers were growing hoarse, everyone was tired and giddy, telling jokes ("stolen" he'd cry instead of "sold"), moving quickly, passing over items for which no one would bid anything (a large DEC PDP/10 computer was passed over because no one wanted to haul it away).

This late in the game, the auctioneers wanted to close the doors. They started grouping

Copyright © 1985 Tony Bove

things together. Six ADDS terminals were up for sale, and the highest bidder could take all six or just some of them.

As highest bidder I asked for only two of the ADDS terminals, and got some ugly stares from the auctioneer as he barked off the remaining four. They went to someone else for $25 less apiece.

Then came the TeleVideo terminals—eleven of them, nine boxed and two out of the box. I inspected the two out of the box and they were both OK, although one had been opened.

As highest bidder again, I chose two of the TeleVideos. However, I *did not specifically shout the auctioneer's numbers* to the auctioneer, and he wrote down two consecutive numbers. One belonged to a new terminal in a box, but the other belonged to the terminal without a box—the one that had been opened. I inspected it and saw that everything was inside it: the logic board, the power supply, and the CRT electronics. I was not allowed to plug it in and turn it on, but it seemed that everything was there and properly connected.

The rules of most auctions is that you bid on stuff and leave a 10% deposit to hold your purchases. But you can't leave with your merchandise the same day. You must come back the next day, pay in full with a cashier's check or cash, and then cart the stuff away.

When I arrived I was not allowed back into the warehouse until I paid in full. When I got inside and saw the unboxed TeleVideo terminal, I was understandably outraged. The insides were gone—no logic board, no power supply, no CRT circuitry. Just a CRT tube, an enclosure, and a keyboard.

The auctioneers would not let me remove the terminal from my list of purchases. I had already paid for it and had to keep it. I had bought the terminal "as is," although it had certainly changed overnight. One of the auctioneers acknowledged sadly that something must have happened. This employee of the auctioneer happened to have (!) an extra TeleVideo logic board that he gave me. Of course there was no power supply or CRT circuitry, and I doubted that the logic board he gave me would actually work.

So, rule number one: don't buy anything at an auction that can be opened up and stolen from, unless you can cart it out immediately. If nobody would buy such items or lots, auctioneers would have to find ways to protect the merchandise, or throw it out.

Rule number two: if you have to leave it overnight, avoid paying for it until you have it in your hands. If the auctioneer's setup prevents you from seeing it again before you pay for it, use a check accompanied by a letter of credit, because you might be able to cancel the check.

If you find yourself in a similar situation, don't argue with the auctioneers. Most of them have bouncers who don't necessarily think before they act. Take your lesson in sullen silence, not with lumps.

So pay attention to the auctioneer when he yells, "I've got forty can I get fifty I have fifty can I hear sixty I got sixty can I hear a hundred? Got a hundred how about 125 got it how about 150 ok 200? Got it how about 225 got 210 lemme hear 225 got 225 lemme hear 250 got 235 lemme hear 250 got 250! Lemme hear 275? got 275 lemme hear 300? Nobody? C'mon now, this is prime stuff, I got 275 who'll gimme 300? 275 going once, 275 going twice, ARE YA ALL DONE?"

"STOLEN for $275 to the man in the feathered cap..." ⊡

---

6 TIMES FASTER!

SuperFast Software Development Tools

INCREASE YOUR PROGRAMMING EFFICIENCY
with high-performance software development products from SLR Systems.
No other tools approach the speed or flexibility of the SLR Systems line.

"Z80ASM is an extraordinary product...",
Robert Blum, Sept. 84 DDJ

"...in two words, I'd say speed & flexibility",
Edward Joyce, Nov. 84 Microcomputing

ASSEMBLERS

- RMAC/M80 macros
- Nested INCLUDES & conditionals
- 16 char. labels on externals
- Built in cross-reference
- Optional case significance
- Phase/dephase
- Math on external words and bytes
- Define symbols from console
- Generate COM, HEX, SLR-REL, or Microsoft-REL files
- Time & Date in listing
- Over 30 configure options

	Z80ASM -full Zilog Z80	$125
NEW!	Z80ASM+ -all tables virtual	$195
NEW!	SLRMAC -full Intel 8080, with Z80.LIB extensions internal	$125
NEW!	SLRMAC+ -all tables virtual	$195

Z80 CPU, CP/M compatible, or 32K TPA required.

"Z80ASM...a breath of fresh air...",
Computer Language, Feb. 85

VISA MasterCard C.O.D., Check or Money Order Accepted

LINKERS

- Links SLR & M80 format files
- Output HEX or COM file
- Three separate address spaces
- Load map and SID/ZSID .SYM file

SLRNK+ includes:

- All tables overflow to disk
- HEX files do not fill unused space
- Intermodule cross-reference
- EIGHT separate address spaces
- Works with FORTRAN & BASIC
- Generate PRL & SPR files
- Supports manual overlays
- Full 64K output

	SLRNK -fastest memory based	$125
NEW!	SLRNK+ -full featured virtual	$195

Combo Paks available from $199. - $299.

For additional information contact SLR Systems
1-800-833-3061, in PA (412) 282-0864
1622 N. Main St., Butler, PA 16001 Telex 559215

SLR Systems

Circle number 17 on the reader service card.

LETTERS

A Taxing Issue

It seems forever between *User's Guide* deliveries, and then when I get an issue like No. 14, I am terribly disappointed.

In the first week of April, *nobody, but nobody* is still interested in tax returns, or the IRS. Computer owners have enough personal finances that they have already filed, and without a doubt they filed through their own personal CPA firm and are not the least bit interested in buying software and fooling around with it for *less* advantage than a professional provides. Additionally, your reviews came much too late even if they were remotely interested.

I don't know how you go about deciding what you are going to put in the next issue, but you *blew it* with issue 14 as far as I am concerned. Next year will be a completely new IRS program and any software for 1984 will never hack it again. Half of issue 14 was dedicated to a subject that was too late, and quite dead. Consider punishing yourself!!

What seems like six months ago, you published part one of the Turbo Pascal tutorial. Where is part two?

Richard Monroe
Fort Collins, CO

Part two of the Turbo Pascal tutorial has been in the works for awhile, but we held off publishing it for several reasons. We heard that version 3.0 was due, and we wanted to see what changes the new version 3.0 of Turbo Pascal would bring. The original release date of the 3.0 CP/M version was delayed, while the IBM version was released before the CP/M version. Now that the 3.0 CP/M version has started to ship, we can be sure both versions of the product are stable enough to continue our planned series of articles, and we can do an accurate series.

We knew we were in trouble in early December 1984 when we decided to do a tax issue. When we called the tax preparation software companies, they said they would ship their 1984 versions "sometime in January, 1985" after the IRS published tax law changes for 1984. (They have to go through this race for customers every year. This is a tough market to reach, because the timeframe of interest for the average customer is short.)

But that was only the first reason for delay. We all expected the new versions to be available much sooner than they were, but this has been a slow year for the IRS. In late February our system went down, delaying circulation of the March issue by four days, until early March. Finally, the USPS had slower than usual second class mail delivery for this issue, partly due to their new rates and classifications. We apologize for the delay.

We'd like to point out that although relatively few people request extensions in order to delay filing until August, the reviews are useful in another way: they point out flaws in products that are surely to be available in a slightly revised form next year. Use issue 14 to prepare yourself for the next tax season.

Sleaze Patrol Nomination for *User's Guide*

You are hereby Nominated as Sleaze Patrol Subject (Incompetance Corner). I today received my subscription copy of No. 14, featuring several good income tax articles, *one week* before the April 15th filing deadline! Useless for this year, with new tax laws possibly coming out for next.

Ronald L. Sutton
Metairie, LA

Just received Issue 14. Contents very good, but timing would rate zilch!! An issue or two ago the income tax preparation data would have been valuable, but received April 8th, it is worthless for this year!

David Downs
Winston-Salem, NC 27104

We console ourselves with the knowledge that unless tax laws change drastically, which is unlikely, the packages described will be updated again for 1985, and they will be similar enough to the 1984 versions that you'll turn to issue #14 of User's Guide *then.*

We did our taxes the week before they were due, using these articles (half of those people we randomly asked still had to do their taxes the week before, as well). If you keep the required records all year (there are several good money management programs available), doing your taxes doesn't require as much preparation time.

Requests

Magazine is very good. Would like an article on MicroPro's ReportStar.

Rick Williams
Lambertville, NJ

I notice by the letters column in issue 13 that you are planning to do a tutorial on Spellbinder. This would be a most welcome article.

Roger Coe
Navaho, NM

Continued on p. 75

"...the very best collection of utility software I've seen...The Kaypro is advertised as "The Compleat Computer"-but this is the bundle of software that makes that true!"

Dave Gerrold, Profiles Magazine

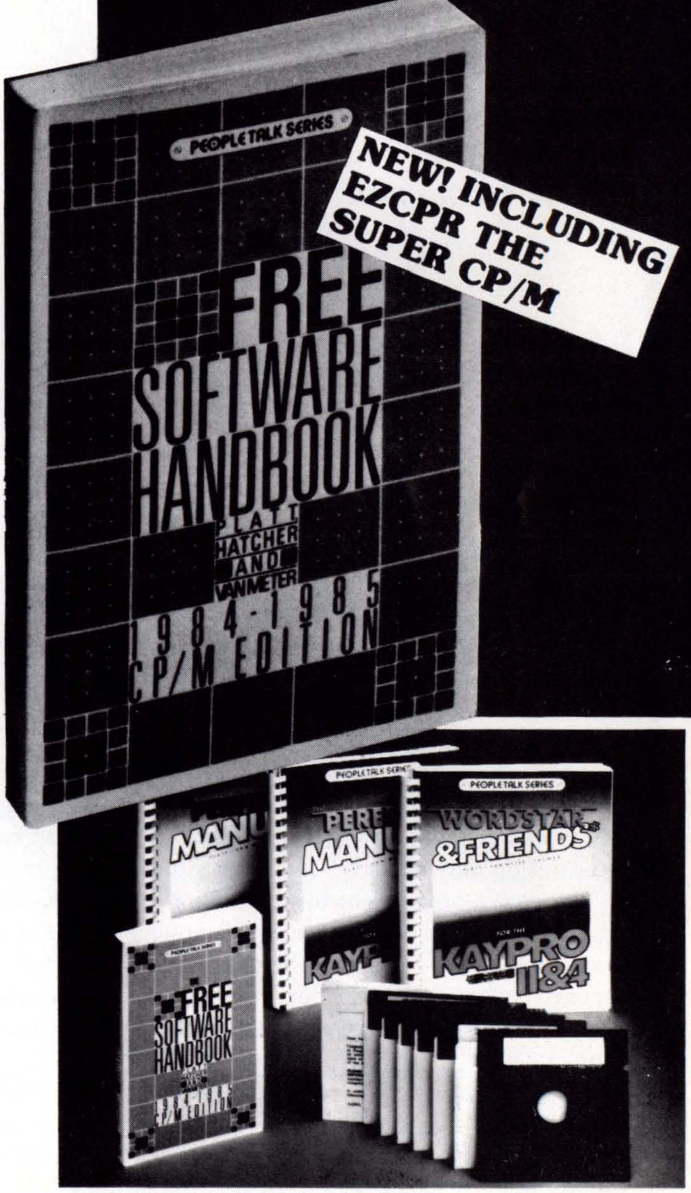
FREE SOFTWARE HANDBOOK ONLY $17.⁹⁵

(Handbook plus software disks $62.95. Disks only $54.95.)

Features the top programs in public domain

- What they are
- When they're useful
- How to acquire them
- How they work

*If ordering disks please specify machine and disk format.



Also from PeopleTalk:

Wordstar and Friends for Kaypro 2 & 4	**$19.95**
Perfect Manual for the Kaypro 2,	**$17.95**
Perfect Manual for the Kaypro 4,	**$17.95**

Supercharge your Kaypro with Smartkey!™

Only $44.95 to customize your computer. Program your own keyboard to turn the most-used commands into ONE KEYSTROKE.

PERFECT WRITER ON THE KAYPRO™

Easy-to-read tutorial for Perfect Writer users. Unleash the power of Writer with this handy reference book as only PeopleTalk could produce it!

- Exceptional type-this-and-you-get that reference section lets you look up a command quickly and easily
- Focuses on all Kaypro models

All this for only **$17.95**

DISKETTES! Special 25 disk prepacks

Single-Sided: **$1.25** each **$31.25** Box

Double-Sided: **$1.45** each **$36.25** Box

Includes warranty, envelopes & labels

CHECKS AND BALANCES

"The sky opened with a crack of thunder ... My checkbook had balanced ... It hadn't balanced in 17 years. If anyone had told me that I would be balancing my checkbook and enjoying it, I'd have suggested he was a candidate for a rubber room!" Dave Gerrold Profiles

The incredibly easy-to-use accounting package for personal and business use.

Super-low price **$54.95**

MONEY-BACK GUARANTEE

See your dealer or order direct:

PeopleTalk Associates Inc.

P.O. Box 863652- C
Plano, TX 75086
1-800-PT BOOKS

Please include product price in U.S. Dollars + $3 shipping and handling ($10 air mail outside U.S.) Texas residents add 6% tax.

Circle number 13 on the reader service card.

Patch It Up

Have you ever wondered what "patching" means, and what it can for you? Patching is explained, with common WordStar® patches.

Sahnta Pannutti

by Richard Drakeford

Suppose you heard at a user group that you could improve your copy of WordStar by installing "a set of patches." You might want to speed up the response, install a special printer feature, make use of user-defined printer controls, or simply change WordStar's default settings so that it comes up with justification off or help levels set to a lower number.

For these and many other reasons you may want to "patch" your copy of WordStar. To do so, you need the "patching information" and a quick course on patching itself. You can patch any program, not just WordStar, so the information may be useful for other things.

To *patch* a program means to modify the existing program directly without bothering to completely rewrite and reassemble a "new" version. In patching, existing code is overwritten at specific memory locations to achieve specific effects, such as fixing a tiny bug or changing a single variable.

In most cases you have no choice in the matter — you must patch a program, rather than reassemble it, because you *can't* reassemble it. For example, to fix up WordStar, you are not provided the *source code*, which you would need to reassemble the program (the source code for most commercial programs is usually proprietary and not available).

Patching, arguably an expedient but inelegant way to change a program, is not in any way dishonorable. In fact, one of the great advantages of WordStar over some other word processing programs is that WordStar contains many areas purposefully left to be patched by the user by use of the INSTALL program, thus permitting a high degree of customization of features. And even after it is installed, the WordStar program is not locked in concrete but is still amenable to continued patching.

Like other CP/M programs, the assembled WordStar program you purchased consists of numerous machine language instructions which are loaded sequentially into your computer's memory beginning at *address* 0100H ("H" is for hexadecimal, and addresses are explained below). All CP/M programs load at this address in memory.

To patch such a program, you load it into memory, move to a particular address in memory, and physically change the instruction or value at that address. Then you use the SAVE command to save that area of memory (containing the modified program) onto a disk.

WordStar is a registered trademark of MicroPro Int.

Hand Patching Byte by Byte

To correctly install a patch, two things must be known:

1) The correct address in memory at which to patch.

2) The exact value to be inserted at that location.

You can access specific locations at known addresses in memory by using the special program INSTALL.COM which came with your WordStar or by using the DDT.COM program which is one of the general CP/M utilities delivered with CP/M. Either will let you go directly to the location by using its address number. Once at the location, both INSTALL and DDT will let you overwrite the current contents with the new desired value. Call this procedure "hand patching."

Concerning the practicalities of patching by rote: If you don't need a deep understanding of what the heck goes on in WordStar when it is operating, then it is perfectly okay to install patches by rote. Perhaps you can't afford the time or just don't care to invest the hours required to gain esoteric knowledge about internal WordStar affairs. Thus, if some expert has designed a patch and you find out what it is, i.e., where to install new values and what those values are, and you desire to achieve the effects that are the result of that patch, patch away on faith. The results will be the same for you as for someone who knows in complete detail what is going on in the bowels of the computer. You can take two aspirin without knowing all about blood chemistry. You can change spark plugs without knowing all the theory behind internal combustion engines.

Labels: A Place By Any Other Name...

Besides going to an address by its numerical (usually a hex number) designation, there is another method of getting to a place to patch: by reference to a *name* for the location. Like a house in a ritzy neighborhood, an address may be known by a name in addition to a number. 1411 Seaside Heights could be known alternatively as "Seven Gables." Many computer programs may have locations identified by names, known commonly as *labels*.

These labels are shadows remaining from the source code. Before it was assembled into a monster collection of machine language code, the assembly language source WordStar program instruction segments had names which roughly indicated their function. During assembly and conversion to machine language the labels themselves disappeared but the functions (values) of each label were stored at specific addresses (somewhat arbitrary, depending on the assembly process). If you know the address of a code segment to do a certain function after assembly, it is often convenient to refer to that location by its functional label.

In fact, it is so convenient that INSTALL programs (used to modify an assembled program) often have an internal index to keep track of which locations in the assembled program belong to which labels. Thus, if you know the label of the function or variable you want to get to, using such a specially rigged INSTALL program as a tool you can go to the location by entering the label name rather than the numerical address. If you know the label and use the right INSTALL program which can find the location by label name, you do not ever need to know the numerical address.

To repeat, labels are a convenient way to refer to the beginning address of a group of machine language instructions or to refer to a variable location. As a specific example, the label `BLDSTR` refers to a variable location which contains the value which WordStar will use to determine how many times to make the printer strike the paper when boldface printing is called for.

WordStar contains many areas purposefully left to be patched by the user, thus permitting a high degree of customization of features.

Some other examples of the labels used in the WordStar program are:

`PALT` — The location where you can store the code to be sent to a printer to start printing in an alternate character size. WordStar would refer to this location when it encounters a control character in a file while printing.

`HAVBSY`— A location which can be marked to indicate that WordStar can find out if the printer is ready for information during concurrent printing and editing. WordStar will then operate differently than it would if it couldn't check on the printer.

`ERAEOL`— Where code which gets the screen to instantly erase from the cursor position to the end of line (varies depending on your system) can be installed.

There are hundreds of labels associated with WordStar. As you can see, although some are cryptic, labels often have a kind of mnemonic association with their actual function.

Labels for patch locations are also a convenience for humans to use when talking about patches. Often you will find references to label locations published in the human-readable English language as "indexes" to WordStar in magazine articles and tip sheets, or in text files. Such indexes will give a label, its function, the number of its address in the assembled WordStar program, and a hexadecimal value that is recommended for entry at that location to make WordStar perform in a certain way.

As an example, you might run across a tip phrased like this: "Patch the value of `DEL3` to a higher number such as 1Fh in order to delay the appearance of prefix menus after you are familiar with most commands."

Conveniently, even though addresses have changed from version to version, the labels have remained the same for different CP/M versions of WordStar. If you know a label for a function in WordStar 2.26, you can use the same label to get to the location of that function in WordStar 3.0 or 3.3. MicroPro was quite good about keeping the continuity of labels from one version of WordStar to another, even though the addresses necessarily changed as the program was re-written and re-assembled.

To recap, hand patching is direct overwriting at specific locations in memory after a program has been loaded into memory. There are several ways to get to a WordStar patch location: 1) go directly to the hexadecimal number address if known (you can use DDT.COM or INSTALL.COM) and 2) refer to a functional label name and go to the label via INSTALL which knows where the labels are, or 3) look up the label in an English index to label addresses and then go to the address number by DDT or INSTALL. After writing new values directly into various locations, the program is saved in its modified (patched) form.

Batches of Patches

So far we have been talking about patching individual WordStar locations one by one — hand patching. The ultimate in patching proficiency, however, is to accomplish wholesale patches by writing an assembly language program which, when run, modifies numerous WordStar locations all at once. Automatic patching! Programs which modify WordStar are indeed feasible and in fact many such programs have been written. Some are commercial and proprietary but quite a few have been donated to the public domain by their authors and are free for the asking.

Although most WordStar users are not ready to write a WordStar modifying program, many users are capable of running an already written program which modifies WordStar in some desirable way — say to install the collection of patches which lets WordStar control a particular printer. Such programs may be obtained from other WordStar users, from user clubs and user group libraries, and often from electronic bulletin board systems accessible by telephone. Once you obtain such a program, it is a simple matter to give the CP/M command which runs the program and patches a WordStar for you.

As an example of what can be done along this line, there is a reasonably priced commercial printer installer program which will automatically modify WordStar to work with your selection from a menu of any one of a dozen printers. Plus this program installs other features. (It should be noted

```
;                               PATCH.ASM
;                       for WordStar Version 2.26
;This is a "fill in the blanks" template file for patching printer
;control codes into the WordStar printer control patching areas.
;Consult printer manual to determine exact values which go into
;each control code area. First value in each area is the number of
;following bytes to be sent to the printer when the trigger
;control code is encountered in file while printing. The
;subsequent bytes (up to four) are the actual hex codes to be sent
;to the printer. For many printers, the first value will be
;ESCAPE, or hex 1B.  With INSERT off, substitute hex values for
;'nn' indicators in the following program. Then replace all unused
;'nnh' with value 00h before assembly.  Or if a location
;previously patched and no change desired, delete from this file.
;
;Label          Address                 Code which triggers
;               & values
;'nn' indicators in the following program. Then replace all unused
;'nnh' with value 00h before assembly.  OR delete from this file.
;
;Label          Address                 Code which triggers
;               & values
;
        org     06C9h
USR1    db      0nnh                    ; ^Q
        db      0nnh,0nnh,0nnh,0nnh     ;
;
        org     06CEh
USR2    db      0nnh                    ; ^W
        db      0nnh,0nnh,0nnh,0nnh     ;
;
        org      06D3h
USR3    db       0nnh                   ; ^E
        db       0nnh,0nnh,0nnh,0nnh    ;
;
        org      06D8h
USR4    db       0nnh                   ; ^R
        db       0nnh,0nnh,0nnh,0nnh    ;
;
END
;When properly filled in, use CP/M's assembler ASM.COM to assemble this
;file which can be named anything you want but should have the extension
;ASM.  After assembly, the resulting HEX file may be used to "overlay" a
;copy of WordStar version 2.26 and the hex codes you entered at the
;various "blanks" above will be plugged into the correct WS.COM locations
;to make the printer function appropriately when the various control
;codes are entered in a text file.
;
;The procedure is:
;1.  Amend this file with WordStar in the non-document mode.
;2.  Assemble this file to obtain hex file.
;3.  Use DDT to load WS.COM (A>DDT WS.COM)
;4.  Use DDT to overlay hex file on WS.COM
;        a. -INEWFILE.HEX
;        b. -R
;5.  Exit DDT (G0).
;6.  Use CP/M's SAVE command to preserve modified WS.
;        (SAVE 56 WSNEW.COM)
;7.  Test modified WS for correct printer driving operation before
;replacing original WS.
;Or, if available, the MLOAD.COM utility may be used to do steps
;3 through 6 in one simple operation:
;        MLOAD WSNEW.COM=WS.COM,NEWFILE.HEX
;End of WordStar Printer Patching File.
```

Figure 1. *An example of an assembly language program file for positioning patches.*

that INSTALL programs also can automatically "spray" batches of patches into a program if the correct menu choices are available. However, no INSTALL can automatically cover every contingency that may arise.)

Of course, if you use an automatic WordStar patching program to install some patches, you will be trusting that the programmer fixed it so his program goes to the right locations and makes the right modifications. Also you may be forced to eat the whole meal the programmer has prepared whether you like everything on the plate or not. Another drawback might be that you may want to install features in two programs but may not be able to because they have been written so that they conflict with each other.

And since the WordStar feature choices are numerous and there are many different computer/ printer/ system combinations, there is a good chance of not finding an existing program which patches WordStar on your system exactly the way you want it to be patched. Therefore you may be interested in another method which allows somewhat more flexibility of choice.

Overlay On MacDuff

The final method of patching up for discussion is not quite assembly language programming but does use an assembler to install a selected batch of patches. This is the hybrid "overlay" method, which yields a good compromise between freedom of choice of patches and automation of the actual installation of new values. You write a very simple assembly language program which does nothing more than designate certain values for certain specific locations, as many or as few as you like. Put these in a program file named, for example, PATCH.ASM (the name can be anything you want but the extension must be ".ASM").

Assemble the "program" and save the resulting ".HEX" type file. You load your WordStar ".COM" program into memory with DDT, and then "overlay" the prepared ".HEX" file (read and merge it with the WS.COM image already in memory) to modify the designated locations. Use the DDT `I` and `R` commands to accomplish this. Then save your newly touched up WordStar, perhaps under a new name such as WSTEST.COM. (Consult a CP/M tutorial to determine how big a chunk of memory to save.)

The ultimate in patching proficiency is to accomplish wholesale patches by writing an assembly language program which, when run, modifies numerous WordStar locations all at once. Automatic patching!

This program patches the eight different locations necessary to control the double wide printing of an Epson MX-80 printer. When the patched WordStar encounters the control character `^W` while printing a file, it sends the correct sequence to the printer in order to turn on the double-wide character font. And when it encounters `^Q`, it turns off double-wide characters.

```
        org     06C9h
USR1    db      003h                    ; ^Q
        db      01Bh,057h,000h          ;
;
        org     06CEh
USR2    db      003h                    ; ^W
        db      01Bh,057h,001h          ;
END
```

Figure 2. *A small assembly language program file "filled in" (with extensive comments stripped out).*

The minimum such patching program would consist of the following:

```
org nnnnh↵
db  0nnh↵
```

— where `org` gives the hexadecimal address of the location to be patched and `db` gives the specific hexadecimal value to be placed there. (The leading `0` is required by the assembler to unambiguously identify the following value as a number since `db` may also include alpha characters.) Yes, such a tiny "program" can be assembled using ASM.COM (the CP/M assembler), and the resulting ".HEX" file can be overlayed into an existing WS.COM (installed WordStar program).

More than one sequential byte might be indicated, as follows:

```
org 06CB↵
  db 002h,01Bh,022h↵
```

— indicates that the three locations starting at address `06CB` are to be patched with values `02`, `1B`, and `22` (hexadecimal notation).

Quite likely, the ".ASM" file you create will consist of several locations to be patched with several bytes starting at each location, as follows:

```
org 06CBh↵
  db 002h,01Bh,022h↵
org 06D3h↵
  db 002h,01Bh,021h↵
```

You may want to include labels and comments to yourself so that you can easily go back and change things later. A heavily commented PATCH.ASM file which you can use as a "template" to patch several locations of WordStar Version 2.26 is shown in figure 1. A minimum configuration of the same program as it might appear after filling in and stripping out comments is included as figure 2.

One good reason for doing patches by this overlay method (rather than modifying byte by byte with INSTALL or DDT) is that you can make ready all your changes in one file and proof-read for correctness before actually modifying your working program. Whereas if you type them directly into the program byte by byte using INSTALL or DDT, you risk human error that can go undetected until the buggy program is run.

To assure yourself that this kind of minimal assembly language patching file does assemble properly and that you know how to do it, you can use the template file for experimentation. Probably you will want to eliminate the comments and some of the addresses and patch identifiable "nonsense" values (such as 0FFh) into various addresses. Use WordStar's `N` (non-document) mode to create the file. (There is something recursively poetic in using WordStar to modify itself.)

To assemble your PATCH.ASM, at the CP/M `A>` prompt type:

```
A>ASM PATCH↵
```

Next load your WordStar with DDT.

```
A>DDT WS.COM↵
```

Insert your PATCH.HEX file; `I` loads the filename into memory, `R` loads and merges the file's contents with the WordStar file (WS.COM) already in memory:

```
-IPATCH.HEX↵
-R↵
```

Exit DDT and save your new WordStar:

```
-g0↵
```

```
A>SAVE 56 WSTEST.COM↵
```

To verify that your patches took place, use DDT again to load the new WordStar and check the locations you wanted to change.

```
A>DDT WSTEST.COM↵
```

The following DDT command can be used to display memory contents, and allows you to alter values.

```
-Snnnn↵
```

Press Return again to keep the value displayed, or type a new value and then Return, or a period (.) to stop the `S` command.

If you want to display memory contents without making changes, use the `D` command:

```
-Dnnnn↵
```

If the patching was effective, at the addresses given in your `org` statements, your WordStar should now hold the values you listed in your `db` statements.

Semi-automatic Overlay

Now that you know the patching procedure just described, you will appreciate that there is an easier way to accomplish the same thing. There is available in the public domain, free for anyone's use, a "linking loader" named MLOAD.COM. This is the work of Mr. Ronald G. Fowler who has freely donated his program for the use of the computing public.

MLOAD.COM is available on many telephone RCP/M's and in the disk libraries of various user's clubs. The latest version I have seen is MLOAD21 (version 2.1), often distributed with the public domain communications program MEX because MLOAD is handy for combining overlays for specific computers with the generic MEX.COM.

I find that MLOAD will automatically "spray" patches into WordStar addresses after you write (using WordStar's `N` for non-document mode) a very simple PATCH.ASM file (just `org`s and `db`s as previously described) and make a ".HEX" file out of it.

The whole procedure:

```
A>ASM PATCH↵
```

— (previously written PATCH.ASM file)

```
A>MLOAD WSNEW.COM=WS.COM, PATCH.HEX↵
```

The first command assembles your file PATCH.ASM and makes up a PATCH.HEX file (as well as a PATCH.PRN file, extraneous to our purpose here, so not discussed). The second command makes up a new file, WSNEW.COM by combining your regular WS.COM and the newly conceived PATCH.HEX.

The same thing can be accomplished with DDT commands and a `SAVE` as described earlier, but MLOAD just does it, and fast. Many people owe thanks to Ronald G. Fowler for providing this nifty tool to the public domain.

Modifying a program by this "overlay" or "insert" method simplifies patching a lot and is a more flexible procedure than using someone else's pre-canned set of patches which might not be just what you want.

Conclusion

The foregoing advice and instructions should serve as an introduction to the idea of patching. Applying a patch avoids the bother (or impossible task in the case of proprietary programs) of going back to the original assembly language source code and making modifications in assembly language by writing within the entire major program. Rather than doing that and then re-assembling the entire program from scratch, the modifications are applied to the existing ".COM" file by direct overwrite.

Several methods have been discussed for applying patches. They are:

1) The use of proprietary INSTALL programs, and the use of DDT.COM to do hand patching, byte by byte.

2) The use of canned programs which apply batches of specific pre-configured patches.

3) A simple overlay method where your patches are written up beforehand and assembled into a ".HEX" file which is overlaid on the main program.

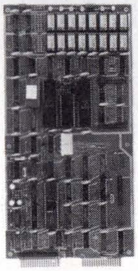
Either DDT.COM or the more convenient MLOAD.COM may be used to integrate the overlay patch file with the main program which gets patched. □

MicroMate

CP/M 3.0 COMPUTER

MicroMate
PMC

System or Board Level to suit your application



- Use with RS 232C terminals
- Thousands delivered — in use
- 128KB RAM and autoboot ROM
- Interfaces four 400 KB disks
- 10-20MB hard drive add-on
- Abundant software available
- Excellent price/performance

Personal Micro Computers, Inc.

PMC
275 Santa Ana Court
Sunnyvale, CA 94086
Phone 408-737-8444

Circle number 14 on the reader service card.

AN INVITATION TO JOIN KUG

KUG invites you to join the more than 15,000 computer user members from around the world who have one common thought. . . .

FINDING A HELPING HAND!

KUG is now in its third year of offering services through its bi-monthly newsletter, **KUGRAM®**. **KUGRAM®** is a 32-page publication featuring articles such as:

CP/M STUFF
The U-Do-It CP/M tutorial

MS/DOS STUFF
The U-Do-It MS/DOS tutorial

WORDSTAR by CHIANG
Working with WordStar the easy way

BEST of KUGBB
Tips from our National Bulletin Board

SYSOP'S CORNER
Information on using modems & software

NATIONAL KUG DISKS
Descriptions of our latest P.D. library

LIBRARY
More than 50 P.D. disks and growing

IN RESPONSE
Answers to member's questions

VENDORS
Listings of the latest computer products

BOX 100
Helpful hints from around the world

Plus reviews, programs, software tips and tutorials.

For word processing members **KUG** offers configurations for most printers. **KUG** also maintains a national **KUG Bulletin Board** system and an on-line Special Interest Group on **CompuServe**.

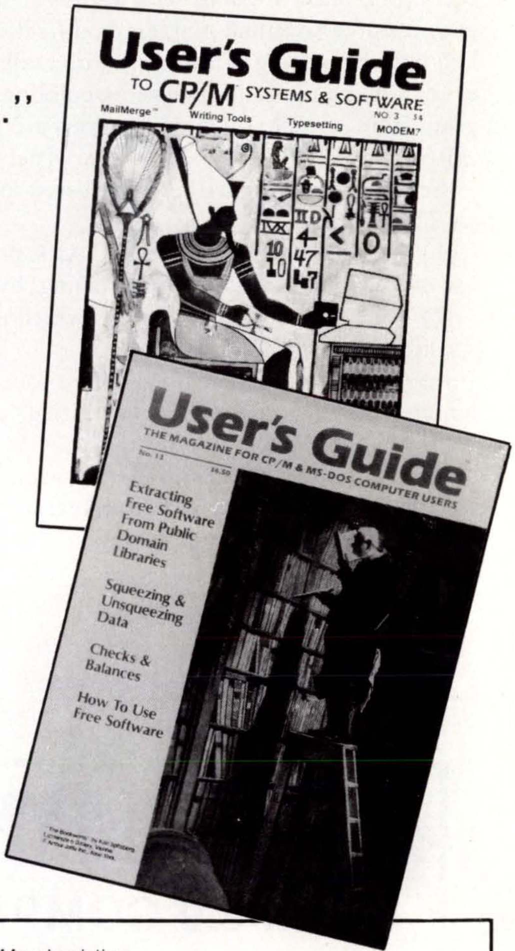
To become a member of **KUG**, send your name, address, phone number and the make and model of your computer along with a check for **$15.00** (U.S. annual fee) to:

KUG....BOX 100G....MALVERNE, NY....11565

............................(516) 746-0056.............................

Circle number 7 on the reader service card.

We Explain Software Packages,
Then We Teach You How To Use Them.
User's Guide is the Magazine of Tutorials.


User's Guide
TO CP/M SYSTEMS & SOFTWARE
MailMerge
Writing Tools
Typesetting
NO. 3
MODEM7
User's Guide
THE MAGAZINE FOR CP/M & MS-DOS COMPUTER USERS
Extracting Free Software From Public Domain Libraries
Squeezing & Unsqueezing Data
Checks & Balances
How To Use Free Software

> "...worth more than it costs."
>
> Jerry Pournelle, ***BYTE***

User's Guide® magazine helps you use CP/M® and MS-DOS® application software on your personal or multi-user computer. For the low cost of a magazine subscription ($21), you get six issues stocked with tutorials and software evaluations ($4.50 each on the newsstand). No fluff, just direct, readable "how to use" information for users of computers that run CP/M software.

If you have a CP/M or MS-DOS computer, you need to know where to find good software and accessories. ***User's Guide*** has in-depth evaluations of commercial ***and free public domain*** software, so that you can choose the best buy.

Don't spend hundreds on training packages. ***User's Guide*** teaches you how to use the most popular programs on the market, such as WordStar®, SuperCalc®, dBASE II® and Modem7. You learn at your own pace.

Editors Tony Bove and Cheryl Rhodes are the acclaimed writers of several computer books on CP/M and WordStar. Contributing editors and columnists include other great writers who use computers extensively, such as Arthur Naiman, Steve Rosenthal, Jonathan Sachs and Kelly Smith. The writing is crisp, intelligent and informative, without an overuse of jargon.

User's Guide also keeps up with the changing personal computer market, with informative articles about communicating with other systems and transferring programs and data.

NO RISK OFFER:

Try ***User's Guide*** today. Fill in your name and address on the card next to this ad. Check the box marked "bill me" and drop in the mail; if you don't like the first issue, write "CANCEL" on your bill and keep it (or better yet — pass it on to someone who can use it). If you think you'll want it, save $3 and send in your payment with the card or coupon.

Try ***User's Guide*** as a training tool. Make your computer investment profitable.

User's Guide
P.O. Box 5245
Redwood City, CA 94063

Indicate Offer and Length of your *User's Guide* subscription:

Take Advantage of the Low Price for Enclosing Payment Now!	Or, try our no-risk bill-me offer and cancel if not satisfied!
☐ I enclose **$21** for 6 issues	☐ Bill me **$25** for 6 issues
☐ I enclose **$40** for 12 issues	☐ Bill me **$45** for 12 issues

☐ **Foreign:** (prepaid US $$): Canada & Mexico $27/six issues, overseas $40/six issues (airmail).

Payment by charge card: ☐ Visa ☐ Mastercard Exp. date ____________
Phone: 415-364-0108.
Card # ____________ Interbank # ____________
Signature required: ____________
Name ____________
Company ____________ Title ____________
Address ____________
City, State, Zip ____________
☐ Check if you do not want promotional mailings.

Make checks payable to ***User's Guide***. Offer expires 12/31/85
Subscription Dept.: P.O. Box 5245, Redwood City, CA 94063.
Please allow 6-8 weeks for processing.

UG15a

• ***User's Guide*** is published by TUG Inc., with no strings attached to manufacturers or software distributors.

User's Guide is a trademark of TUG Inc. CP/M is a registered trademark of Digital Research, Inc. WordStar is a registered trademark of MicroPro Intl. SuperCalc is a trademark of Sorcim. dBASE II is a trademark of Ashton-Tate. MS-DOS is a trademark of Microsoft.

Circle number 19 on the reader service card.

FOG

The **F**IRST **O**SBORNE **G**ROUP (**FOG**) has been formed as a User's Group for persons using or interested in portable computers and/or the CP/M disk operating system with related software. In December of 1984, there were over 15,000 members from around the world. Most attend local group meetings at over 300 locations.

Computer systems owned or used by members include the Osborne 1 (single and double density), the Osborne Executive 1, all models of the Morrow MicroDecision, the Zorba, all models of the KayPro, several MicroMates, and many more. Special interest groups organized to augment a network of local group meetings include dBase II, Ham radio operators, Personal Pearl.

FOG was started in October of 1981 by a small band of early buyers of the Osborne 1. The primary purpose was to organize a library of public domain software to run on the Osborne 1. A newsletter was quickly started to act as a focal point for the group's activities. The large number of excellent contributions to both the library and the newsletter has produced a library of 230 disks (as of December, 1984) and a nicely typeset (70 or more pages) monthly publication. All back issues of the ***FOGHORN*** are available for a nominal fee which includes shipping in the U.S. Contributions are currently being solicited for bulletin board systems in addition to the thirty currently in operation. Most systems accept both 300 baud and 1200 baud. Phone numbers for the first systems are:

System #1 — Daly City, CA	(415) 755-2030	24hrs/7 days
System #2 — Vancouver, BC	(604) 596-0314	24hrs/7 days
System #3 — Daly City, CA	(415) 992-8542	24hrs/7 days

Each of these systems will have a file listing phone numbers for the other systems.

While the meetings are organized on a local basis, over 130 of these local groups have joined the FOG network, thus increasing the sharing of information, tips, problems and so on. Those local groups which opt to formally join the FOG network receive a portion of local member dues to assist with the cost of maintaining a local copy of the disk library.

The FOG library is currently maintained on the Osborne 1 single density format but separate libraries are being established for the other computer formats. The library files are carefully screened and divided into category types (utilities, games, applications, and computer languages are the four major categories). Programs which contain run or other errors are put into the hacker section so interested members can fix them and resubmit for inclusion in the correct section. Items which do not fit into one of these categories are in the miscellaneous section. A catalog and descriptions of all the discs is maintained in the library section.

Dues in FOG are $24.00 per year. This entitles each member to a copy of the ***FOGHORN*** each month as well as access to the disk library. Local group meetings are open to the public without charge although access to the disk library is restricted to the membership. The FOG library contains only public domain software. Piracy (the copying of proprietary software) is strongly condemned.

In the United States, the ***FOGHORN*** is normally mailed by non-profit bulk mail. (FOG is a corporation in the state of California and has obtained its non-profit, tax exempt status from both the state and federal governments.) For those members who live out of the country or who prefer first class delivery of their ***FOGHORN***, additional postage must be added to the annual dues. See the chart below for details.

If you are interested in joining a self-help organization to increase you knowledge and the use of your computer, use the application below (or a copy of it). Generally, memberships received at the FOG office prior to the 15th of the month are entered in time to receive the next month's ***FOGHORN***.

If you know of a local group which might be interested in joining the FOG network, please send all details (meeting dates and places, officers, and how interested local computer owners can join). We will send you an information packet on becoming an Affiliated Member Organization.

For your records, the address of FOG is P. O. Box 3474, Daly City, CA, 94015-0474. Please allow at least two months for the arrival of your first ***FOGHORN*** since bulk mail can take as much as nine weeks. (The post office says that it should only take about three weeks for non-profit bulk mail but some members on the East Coast have experienced longer delays.) A membership card will be processed within two weeks of the receipt of your dues.

ADDITIONAL POSTAL CHARGE CHART

Please add the appropriate amount to your dues payment.

Canada & Mexico (First Class Airmail delivery)	ADD...$ 9.00
Members with U.S. addresses who prefer First Class delivery to bulk mail	ADD...$ 9.00
Central & South America, Caribbean, & Europe (Airmail First Class delivery)	ADD...$12.00
Asia, Africa, & Far East (Airmail First Class)	ADD...$15.00
Out of North America preferring surface mail — delivery not guaranteed	ADD...$ 6.00

CUT HERE

NAME: ____________________

Company (if part of mailing address): ____________________

ADDRESS: ____________________

CITY: ____________________

STATE: __________ ZIP or MAIL CODE: __________ COUNTRY: __________

HOME & WORK PHONES: ____________________

MEMBER OF WHICH LOCAL GROUP? ____________________

COMPUTER TYPE? ____________________

MODEM TYPE? __________ PRINTER TYPE? __________

INTERESTS? ____________________

Send this completed application **AND** your payment to:

FIRST **O**SBORNE **G**ROUP
P. O. Box 3474
Daly City, CA 94015-0474
United States of America
Phone: (415) 755-4140

Be sure to include any required postage surcharge per chart above.
Your membership card will be sent by 1st class mail.

Circle number 5 on the reader service card.

CP/M-80 •
CP/M-86 •
PC-DOS •
MS-DOS •

WordStar Games Your Mother Never Taught You: Part 3

Word processing techniques for handling large WordStar® documents.

by Jonathan Sachs

WordStar's popularity seems to be most pronounced among "production" writers: those who use computers to turn out large volumes of text, day after day. WordStar may lack some of the convenience and features that appeal to casual users and writers of correspondence, but for entering and editing documents efficiently, it can't be beat.

If you're a "production" writer, chances are that some part of your work involves editing very large documents. This article describes editing techniques that can be particularly helpful in that kind of work.

How Large Is Large?

You can edit a document of any size with any word processing program by dividing the document into files that are small enough for the program to handle. This is an inefficient way to edit a large document, though. If you need to find a piece of text, and you aren't sure where it is, you may have to search several files to find it. And if you need to make a change that affects a large part of the document, you must load and save many files to do so. Thus, it's desirable to store a large document in the largest files you can — and thus in the fewest files you can.

If your computer runs CP/M-80, from 52K (kilobytes) to 56K of RAM typically is available for running programs. Most word processing programs occupy 20K to 30K of this space, leaving something like 22K to 36K of space for text buffers. This space can hold from 10 to 16 pages of average single-spaced text. If your computer runs a 16-bit operating system like CP/M-86 or MS-DOS, it probably has more than 64K of RAM, and the effective limit on file size will be greater.)

WordStar is a "virtual memory" editor: it can edit files that are too large to fit in RAM all at one time. You pay a price for that capability: as you move around in a large file, your computer must write some parts of the file back to the disk to read other parts in. The larger the file, the smaller the fraction of it that you can edit without such disk operations.

If your computer has only floppy disk drives, disk operations will be slow enough to put an effective limit on the size of the file you can edit. "Effective limit" means that you can edit larger files if you insist, but you generally won't want to try. If you do, you'll spend too much time waiting for the disk drive to catch up with you. The whole point of using a computer is to speed up your work, and if you use it in a way that makes you wait all the time, you're defeating yourself.

The effective limit on the size of a text file is somewhere between the size that will fit in your computer's RAM (along with CP/M and WordStar), and about three times that size. The exact limit will depend on your working habits: whether you skip around in a document a lot, or tend to work on one part at a time. It also depends on the speed of your floppy disk drives. Watch for the effective limit as your documents grow; you'll know when you've reached it.

Raising the File Size Limit

Storing text files on a hard disk will speed up disk operations, and so will raise the effective limit on file size.

WordStar reads a file much more slowly when moving backward than when moving forward. If you have to move backward in a large file, you'll often find it more efficient to save the file (enter `^KS`), which re-positions WordStar at the beginning of the file; then move forward to the place you want. This technique has the additional benefit of saving your file on disk from time to time.

Most 16-bit operating systems can use a *RAM disk:* a program that makes part of your computer's RAM act like a disk drive. The RAM disk's advantage over a real disk is speed: since it has no moving parts, it "reads" and "writes" almost instantly. You can increase the effective limit on file size by copying a large file to a RAM disk, editing the copy, and then copying the file back when you're done. (Since everything in a RAM disk is lost when the power is turned off, you cannot use it for permanent file storage.)

Some CP/M-80 systems have extra-large RAMs and can use a RAM disk, even though CP/M-80 programs can use no more than 64K of RAM directly.

What's "Too Large?"

Even if your computer makes all of a large file instantly available to WordStar, other problems limit the size of file you can edit. When you exceed the limit, you'll find that WordStar takes a long time to move from one part of the file to another. Thus, storing a document in a few too-large files can make for problems, just as can storing it in too many small files.

So what's "too large?"

If your computer uses small floppy disks (under 250K or so), no file should be larger than one-third of the capacity of a disk. When files get larger than that, WordStar has to start deleting ".BAK" files to save new versions of documents. That's not a safe way to work.

If your disk capacity is not a factor, the most useful rule is that *a file is too large when it takes you an inconveniently long time to move from one point in the file to another.* Many of the techniques described below require you to have access to all parts of your file at all times; if you can't get that access quickly you'll have to choose between waiting a lot and giving up these techniques, and your control over the work you do will suffer.

My own experience is that even with a fast hard disk, a file larger than 80K or so is inconvenient to work with, and a file larger than 120K is very awkward indeed. This is about 35 to 50 pages of single-spaced text, and so is a generous amount for a story or article, or one chapter of a book. Larger documents generally should be split into two or more files; they usually have natural stopping points (*e.g.*, ends of chapters) that can be used as points of division.

Where Did I Write That?
The "Find" Command

WordStar's "Find" command, `^QF`, is a powerful tool for finding your way around a large file. Most WordStar users know how to use `^QF`, but few are aware of all its capabilities.

This section introduces `^QF` for WordStar users who are not familiar with it. The following sections describe some of its advanced features.

Suppose you're writing a story, and you want to work on a particular scene; you don't know exactly where the scene is, but you remember that it begins with the words "On Wednesday morning. . ."

If the word "Wednesday" does not appear frequently in your story, it makes a good *find string.* Enter `^QF`. WordStar prompts you for the string you want to find:

```
FIND? _
```

Enter the find string:

```
FIND? Wednesday↵
```

WordStar prompts you for some *options* that will influence the way it searches for the find string:

```
FIND? Wednesday↵  OPTIONS? (? FOR INFO) _
```

To do a plain vanilla "Find," just press Return (↵). WordStar searches forward from the position of the cursor, and stops at the end of the first occurrence of the string that it finds. If WordStar doesn't find the find string, it moves the cursor to the end of the file and displays a message like this:

```
*** NOT FOUND: ''Wednesday'' *** Press ESCAPE Key
```

Then WordStar "freezes" — it won't continue until you press the ESCape key. This protects you from making inappropriate changes to your file if you typed ahead while `^QF` was running, in anticipation of finding the find string.

Suppose the first occurrence of the find string is not the one you want? Enter the "Again" command, `^L`, which repeats the last `^QF` command you entered. You can repeat `^L` as many times as necessary, either immediately or after entering any other commands. You can even use it after saving one file and loading another, so long as you don't leave WordStar in between. But when you enter another `^QF` or a "Find and Replace," `^QA`, `^L` changes to repeat this new command.

Editing the Find String

WordStar lets you use several control characters to edit a find string while you are entering it.

Some of these control characters are similar to ordinary text editing commands. For example, `^S` moves the cursor one position backward. It also deletes the last character in the find string, though; thus its function is similar to the function of the DELete (or RUBout) key in ordinary text.

`^D` is the opposite of `^S`; it moves the cursor one position forward, and adds a character to the end of the find string. The character it adds is the one that last occupied that position.

For example, if you enter `^S`, then `^D`, `^D` will reinsert the character that `^S` removed. If you enter `^D` without a prior `^S`, it will add the character that occupied the next position in the last find string you entered (if there was one). Suppose you do a "Find" for **`Wednesday`**, then enter another `^QF` and enter the find string `Thur`. Then you press `^D`. The find string will become `Thure`, because the next character in the last key you entered was an `e`.

`^A` or `^Y` erases the whole find string; either one is equivalent to pressing `^S` until the find string is gone. `^R` re-inserts the whole find string, followed by the remainder of the previous find string, if any; it is equivalent to pressing `^D` until there are no more keys to be added[1].

`^U` *cancels* a "Find" command. Use it when you enter `^QF` and then decide you don't want to find anything. After you enter `^U`, you must press the ESCape key to continue editing.

You can use `^U` to cancel `^QF` even after WordStar has begun searching. This is useful if you realize too late that you misspelled the find string, or entered the wrong one, or don't want to do a `^QF` at all. However, there will be a noticeable delay before WordStar notices the `^U`, and if it does not have far to search, it may finish the `^QF` operation normally.

"Find" Options

The plain vanilla form of `^QF` has the following characteristics:

- It searches forward from the position of the cursor to the end of the file.
- It leaves the cursor at the end of the first occurrence of the find string.
- It considers case; the find string **`day`** won't match the text string `Day`, and *vice versa.*
- It matches any string of consecutive characters; the find string **`day`** will match the text word `day`, but also the `day` in `Sunday`, `daytime`, etc.

You can change each of these characteristics by entering appropriate letters at the `Options?` prompt.

To make `^QF` search *backward* from the position of the cursor to the beginning of the file, use the option `B`:

```
FIND? Wednesday↵  OPTIONS? (? FOR INFO) b↵
```

(Options may be entered in upper or lower case.) `^QF` will leave the cursor at the *beginning* of the first match that it finds in its search.

To make `^QF` stop at the second, third, fourth, etc., occurrence of the find string, enter a number. The following makes `^QF` stop at the third occurrence of the find string as it searches forward:

```
FIND? Wednesday↵  OPTIONS? (? FOR INFO) 3↵
```

To make the `^QF` command ignore case, enter the option `U`. Then the find string **`day`** will match the text string `day`, `Day`, `DAY`, etc.

To make the `^QF` command match only *whole* words, enter the option `W`. Then the find string **`day`** will match the word `day`, but not the `day` in `Wednesday`, `today`, etc. (Punctuation doesn't count as part of a word. Thus the find string **`day`** will match the word `day` written in quotes, followed by punctua-

Option	Mnemonic	"Find"	"Find and replace"
`n`	—	Search for the `n`th occurrence of the find string.	
`B`	Backwards	Search from cursor to beginning of file.	
`G`	Global	not meaningful	Operate on entire file (from start to end unless `B` is also used).
`N`	No prompt	not meaningful	Replace occurrence(s) of the find string without prompting for permission.
`U`	—	Ignore case in searching for the find string.	
`W`	Word	Search for a whole word that matches the find string.	

Table 1. *Options used with the WordStar "Find" and "Find and Replace" commands.*

tion, or hyphenated as in `wedding-day`.)

You can combine "Find" options in any meaningful way, and in any order. The option `2UW`, for example, makes the `^QF` command search forward for the second whole word that matches the find string, ignoring case. The option `WUB` makes `^QF` search backward for a whole word, ignoring case[2].

If you don't want to enter any options, you can avoid the `Options?` prompt by pressing ESCape rather than Return (↵) to end the find string.

You can use `^U` to cancel a `^QF` command at the option prompt, just as you can at the find string prompt.

WordStar can edit files that are too large to fit in memory all at one time. You pay a price for that capability: as you move around in a large file, your computer must write some parts of the file back to the disk to read other parts in. The larger the file, the smaller the fraction of it that you can edit without such disk operations.

Searching for Phrases

`^QF` considers spaces and line ends to be significant. This means that it's seldom safe to search for a phrase like `On Wednesday morning`. `^QF` may miss such a phrase while searching text for any of these reasons[3]:

- The text is justified (the right margin is even), and the words in the text are separated by a different number of spaces than the words in the find string.
- The text contains a soft hyphen (a hyphen inserted during paragraph reforming, which WordStar prints only when it falls at the end of a line). `^QF` wants to match a soft hyphen in the text to a soft hyphen in the find string, even when the hyphen falls within a text line and is therefore invisible in printed output.
- The words in the text fall over a division between lines.

You can deal with the first situation by using a ragged right margin — at least until you're done editing your text.

Similarly, you can deal with the second situation by inserting no soft hyphens in your text until you're done editing. (Enter `^OH`, if necessary, to turn off hyphenation help, and `^B` will stop prompting you to hyphenate long words as you reform paragraphs.)

The third situation is unavoidable. Thus it's better to search for a single uncommon word, and use `^L` until you find the occurrence you want, rather than search for a unique phrase and risk missing it. (If you're searching for a phrase that you *know* is on a single line — a table entry, or the first few words of a paragraph — this restriction does not apply.)

Suppose you want to use `^QF` on a file that already contains soft hyphens? You can use `^QF` to find all the soft hyphens and delete them. First, enter the "Soft Hyphen Entry" command, `^OE`. Now do a `^QF` with a find string consisting of a hyphen; `^QF` will find only soft hyphens, and you can delete them[4]. Enter `^OE` again when you're done, so that hyphens in find strings will match ordinary (hard) hyphens again.

Finding Words in Context

You often can include the context of a word in a find string to find the occurrence you want more efficiently. If you know the word ends a sentence, for example, you can search for the word followed by a period:

```
FIND? Wednesday. ↵
```

It is safe to use this technique together with the `W` option. Think of `W` as meaning "one or more whole words, possibly with punctuation."

If you know a word starts a sentence, capitalize it when you enter the find string:

```
FIND? Wedding ↵
```

Wildcard Character	Mnemonic	Meaning
`^A`	Any	Matches any character.
`^N`	End	Matches a hard or soft carriage return.
`^O`	Other than	Matches any character except the one that follows `^O` in the find string.
`^S`	—	Matches any character except a letter or digit.

Table 2. *Wildcard characters used in WordStar "Find" and "Find and Replace" commands.*

(If you didn't know, and the word wasn't a proper noun, you must use the `U` option to be sure you'll find the word whether it is capitalized or not!)

Similarly, if you know the word begins a paragraph, you can search for `^N` (Control-`N`), followed by whatever number of spaces you use to indent a paragraph, followed by the word:

```
FIND? ^N      Wedding ↵
```

In a find string, `^N` represents a "carriage return" which constitutes a normal end-of-line. It matches either a "hard" carriage return (entered by typing ↵) or "soft" carriage return (inserted by WordStar when re-forming paragraphs).

With thought and practice, you can develop your own repertoire of such techniques. Be cautious, though; these tricks can backfire. For example, if you know a word ends a paragraph, can you search for it like this?

```
FIND? wedding.^N ↵
```

Maybe; maybe not. If you once broke a larger paragraph in two at that point, you may have left a space at the end of the last line after the word "wedding", and the find string won't match the text. You can avoid such problems by using such find strings with care, and by "cleaning up" your text where possible. For example, you can use the "Find and Replace" command, discussed below, to eliminate trailing blanks at the ends of paragraphs, and then the find string shown above will work.

Wildcard Characters

You already know that you can use wildcard characters in command lines to refer to groups of files:

```
A>copy b:*.txt a: ↵          (in MSDOS or PCDOS)
A>pip a:=b:*.txt ↵                   (in CP/M)
```

You can also use wildcard characters in a find string to find one of a class of text strings[5].

Your operating system's wildcard characters are `*` (asterisk) and `?` (question mark). WordStar's wildcard characters, on the other hand, are control characters.

Some `^QF` wildcard characters have editing functions when you're entering a find string, and you must precede them with a `^P` (Control-`P`) to enter them as characters in the find string. For example, `^A` is a wildcard character; but if you type `^A` while entering a find string, it will make WordStar erase the find string and return the cursor to the first position after the prompt. Thus, to put a `^A` in a find string, you must type `^P^A`. (You *can't* type `^PA`; that enters a `^A` in text, but in a find string it just enters the letter `A`.)

The wildcard character `^A` matches *any single character*. For example, if you type this:

```
FIND? b^P^At ↵  OPTIONS? (? FOR INFO) w ↵
```

WordStar will search for the word `b^At`, which matches `bat`, `bet`, `bit`, etc.

You can use `^A` to search for examples of a pattern that occurs in text with variations. For example, suppose a file has many numbered paragraphs that begin with headings like `[1]`, `[2]`, and so on. You could find each of these headings in turn by searching for the find string `[^A]`. (If some of the paragraph numbers had two digits, you would have to find them in a separate operation by searching for `[^A^A]`.)

The wildcard character `^S` matches any character that is not a letter or digit. You can use `^S` to search for the beginning or end of a word. For example, to find all occurrences of `wait`, `waiter`, `waiting`, etc., you could search for the find string `^Swait`. Or suppose you have a tendency to omit the `o` in words like `reaction`. You can search for such misspellings with the find string `tin^S`, and be assured that you won't find such correctly spelled words as `waiting`, `tinder`, etc.

The wildcard character `^N`, mentioned above, matches a hard or soft carriage return. You can use it to find a word known to begin or end a line. You can also use it to find blank lines, such as those that occur between paragraphs (search for `^N^N`) and multiple blank lines, such as occur between sections of text (*e.g.*, search for `^N^N^N` to find two blank lines). You can enter a number as an option to move the cursor to the `n`th carriage return, paragraph end, etc., forward or backward from its current position.

The wildcard character `^O` matches any character *except* the one that follows it in the find string. For example, the find string `q^Ou` matches a `q`, followed by any character except a `u`.

Unlike "Find" options, wildcard characters cannot safely be combined. For example, you might expect that the find string `^O^No` would match any character except a carriage return, followed by an `o`; but it doesn't work.

Entering Control Characters In "Find" Keys

In general, you can search for a control character in a find string just as you would search for any other character. For example, the print control toggle `^B` is used in WordStar text to turn boldfacing on and off[6]. If you have a boldfaced heading such as `^BEarly Theories^B` (as displayed in the text), you can find it this way:

```
FIND? ^P^BEarly Theories^P^B ↵
```

Wildcard characters are exceptions to this rule. What if you entered the following find string while trying to find an *underlined* (with the `^S` toggle — `^SEarly Theories^S`) phrase?

```
FIND? ^P^SEarly Theories^P^S↵
```

You would find any instances of the phrase "Early Theories" surrounded by characters that are not letters or digits, rather than finding ^SEarly Theories^S on the first try. Thus, there's no way to make WordStar search explicitly for a string of text that includes ^A, ^S, ^N, or ^O.

The "Find and Replace" Command

WordStar's "Find and Replace" command, ^QA, is similar to "Find." After it finds an occurrence of a find string, however, it can *replace* the occurrence with another string of text.

For example, suppose you want to replace the next occurrence of Wednesday with Thursday. Enter ^QA. It prompts you for a find string. After you enter the find string, it prompts you for a *replacement string*:

```
FIND? Wednesday↵ REPLACE WITH? _
```

Enter the desired replacement string, and press Return (↵) in response to the Options? prompt:

```
FIND? Wednesday↵ REPLACE WITH? Thursday↵
      OPTIONS? ↵
```

WordStar finds the next occurrence of the find string, and displays the following prompt in the upper right corner of the screen:

```
REPLACE (Y/N): _
```

Enter Y, y, or ^Y to make WordStar replace this occurrence of Wednesday with Thursday. Enter anything else (*e.g.*, N) to end ^QA without changing the text.

Almost everything you've learned about ^QF can be applied to ^QA as well. Since ^QA is more complex, though, it has additional properties that you should know about.

^L applies to ^QA; specifically, it repeats the last ^QF *or* ^QA command you entered.

You can skip the Options? prompt by pressing the ESCape key at the end of the replacement string, just as you can with the ^QF command at the end of the find string.

Wildcard characters may be used in a find string for ^QA, just as they may be used for ^QF. They have no special meaning in a replacement string, though. If you type a ^A into the replacement string, for example, ^QA inserts a ^A into your text.

Using Options With "Find and Replace"

All the options described for ^QF apply to ^QA as well.

^QA with a numeric option finds and replaces the next n occurrences of the find string. It prompts you for the replacement string only once, but gives you the Replace? prompt n times. Thus you can choose to replace or leave each occurrence of the find string that is found.

^QA accepts two options that are meaningless to ^QF. The first one is G, which makes ^QA do a *global* search. That is, it starts at the beginning of the file and continues until it runs out of matches, leaving the cursor at the end of the file. It gives you the Replace? prompt for each match.

The second option is N, for "*no* prompt," which makes ^QA do its replacement(s) without giving the Replace? prompt (you have no choice but to replace). Use this option when you want to replace many occurrences of a find string, and you're *quite sure* that every match WordStar finds should be replaced.

When you use the N option, WordStar pauses momentarily to display each match in the text while it performs the replacement. When there are many matches to be replaced, this pause and display can make ^QA work very slowly. You can eliminate the pause-and-display, thereby speeding up the operation, by entering any key after ^QA starts to work. WordStar will not change the display again until ^QA has finished running, and will usually run much faster.

Caution: When ^QA does finish running, WordStar will interpret that last keystroke you entered as a character or a command. If you don't want that to happen, press the ESCape key, which will have no effect except to suppress the ^QA display of each match.

Some Specific Uses Of "Find and Replace"

"Find and Replace" has a great variety of uses; you will discover the ones that are useful to you as you work. This section only mentions some of the more common ones.

You can use ^QA to delete a string by entering a null replacement string (one zero characters long). Be careful of the deleted word's context, though. If you delete the word alone, you'll leave two or more consecutive spaces:

```
This is the deleted word                    (before)
This is the  word        (after: 2 blanks precede word)
```

When you reform this text, these spaces will be treated as *hard* spaces (they will never be condensed). If you replace the word and the following or preceding space, you will miss occurrences of the word that are preceded or followed by punctuation, or fall at the start of a line. Thus you generally must enter at least three ^QA commands, specifying the same find string in different contexts, to delete every occurrence.

You can eliminate a document's soft hyphens most efficiently by replacing them with "null." First turn on "Soft Hyphen Entry" by typing ^OE (unless it is already on). Then find every soft hyphen that falls at the end of a line, and replace it with null:

```
FIND? -^N↵ REPLACE WITH? ↵
      OPTIONS? gn↵
```

(A line that ends with a hyphen does not have a trailing space, as do other lines that end with soft carriage returns.) Then replace every other soft hyphen with null:

```
FIND? -↵ REPLACE WITH? ↵
         OPTIONS? gn↵
```

Note that, as stated above, the "Soft Hyphen Entry" toggle must be set to `ON` (`^OE`) before you start this operation.

Now, Where Was I?

Being able to keep track of your position is more important when you're editing a large file than when you're editing small files. There's more space to get lost in. WordStar offers several commands that are invaluable for keeping track of your place.

The simplest group of commands are the *place marker* commands. WordStar has ten "place markers," numbered 0 through 9, which you can set at any point in a document. You can move the cursor to any of the place markers at a later time by entering a single command.

The "Set Place Marker" command is `^Kn`, where `n` is the number of the marker. To set place marker 0, for example, enter `^K0`; to set place marker 3, enter `^K3`.

When you set a place marker, a symbol containing the marker number appears before the marker's position in the text:

```
      cursor
        ↓
a symbol_containing the marker               (before)
a symbol‹0›_containing the marker             (after)
           ↑
         cursor
```

The "Move Cursor to Place Marker" command is `^Qn`, where `n` is the number of the marker. For example, to move the cursor to place marker 0, enter `^Q0`.

If you find a place marker distracting, you can remove it from the screen by moving the cursor to the marker (*e.g.*, with `^Qn`) then "setting" the marker again (with `^Kn`). This does not make a marker go away; it just makes the marker invisible. You can still move the cursor to an invisible marker with `^Qn`; when you do so, the marker will become visible again.

Leaving WordStar or returning to its main menu makes place markers go away. When you re-enter your file, you'll have to redefine each place marker before you can use it.

The potential uses of place markers are limited only by your imagination.

If you need to leave one part of a file to work on another part, but intend to return, you can set a place marker in the location you want to come back to.

If you're working on one part of a file while referring to another part, you can set a place marker in each location, and use `^Qn` to skip back and forth between the two. (If the text you want to refer to happens to be in a different file, you can read it into the file you're editing, then delete it when you're done with it.)

WordStar's "Beginning of Block" and "End of Block" markers (set with `^KB` and `^KK`, respectively) are intended primarily for marking the limits of a block of text to be moved, copied, or deleted. You can also use them as you use the numbered place markers, though. `^QB` moves the cursor to the "Beginning of Block" marker; `^QK` moves the cursor to the "End of Block" marker.

Returning To the Source Of a "Find" or "Copy" Command

WordStar has another place marker that is set automatically at the original position of the cursor before a `^QF` (find), `^QA` (replace), `^L` (find or replace again), `^KC` (block copy), or `^KV` (block move) is performed. You can return to the place marker by entering `^QV`.

Here are some examples of how `^QV` may be used.

With "Find":

- You want to see what you said at some point in your file where you remember the word "Wednesday" appears. After doing a `^QF` for **Wednesday**, you may return to your prior location by entering `^QV`. (If the first `Wednesday` you found was not the right one, though, you'd have to go back and set a numbered place marker to return to; `^QV` returns from only the *most recent* `^QF`.)
- You enter a word, then look it up and find you misspelled it. You wonder if you've misspelled the same word elsewhere in your document. You do a backwards `^QF` for the misspelled word, and get WordStar's `NOT FOUND` message. After pressing the EScape key, you may enter `^QV` to return to your working location.

With "Find and Replace":

- You notice a misspelled word near the bottom of your computer's display. The cursor is near the top of the display. Use `^QA` to correct the word; then use `^QV` to return to where you were before you saw the error.

- You want to replace every occurrence of a certain word in a passage of text with another word. You move the cursor to the start of the passage and run `^QA` with a numeric option large enough to cover all occurrences of the word. When WordStar finds the first occurrence of the word *after* the passage in question, enter ^U and press ESCAPE to terminate ^QA; then enter ^QV to return to where you started.

With "Copy" and "Move":

- You want to copy a piece of text from the place where you're now working to another place, then return and make some changes to the original. After performing the copy, enter ^QV to return to the place where the "Beginning of Block" marker was before the copy operation.

- After moving a block of text, you must return to the block's original location to clean up the spacing of the remaining text. ^QV takes you there.

- After moving a block of text, you decide you made a mistake; it belongs where it was before. Enter ^QV, then ^KV, to move it back.

- After moving a block of text, you realize you meant to copy it. Enter ^QV, then ^KC, to make a copy of the block in its original location.

Undoing a Cursor Movement Command

WordStar has one more command that is useful for moving around a file: the "Previous Position" command, ^QP.

WordStar tells you about ^QP in one place: when you enter the "Save File" command, ^KS, you are prompted to enter ^QP "to return cursor to position before save." In fact, you can enter ^QP after *any* command to return the cursor to the position it had before the preceding command was entered.

Here are some examples of how ^QP may be used:

- After making a change within a paragraph, you want to reform the paragraph, then resume editing at the end of the change. Enter ^B, then ^QP.

- Immediately after doing a ^QF or similar command, you want to return to the place where you started. You can enter ^QP more quickly than ^QV, since you enter it with both hands instead of one.

- You enter ^R or ^C to see the part of your text just before or just after the part displayed on the screen. Then you enter ^QP to return to where you were.

- You accidentally enter ^C when you wanted ^V, or ^R when you wanted ^T. ^QP corrects your mistake.

- You keep miscellaneous notes about your work at the end of the file you are working on. You enter ^QC to look at the notes, then ^QP to return to the place where you're working.

- You want to compare two pieces of text, and need to move from one to the other several times. Place the cursor on one piece of text, and enter a ^QF command that moves it to the other; then enter ^QP repeatedly. WordStar will bounce back and forth between the origin and destination of the ^QF command.

^QP is probably the most neglected command in WordStar's repertoire. Everything that it does can be done some other way — with a few more keystrokes. The value of ^QP comes from the fact that it can save you a few keystrokes hundreds of times a day. If you do a lot of editing, those keystrokes add up fast.

Footnotes

1 - You might be inclined to think that ^F performs this function, since the function is the reverse of ^A's function, and ^A and ^F perform opposite functions when you're entering text. If you type a ^F while entering a find string, however, you'll get a display of your current disk's directory! The directory display will go away when you finish the ^QF command.

2 - "Ay, there's the WUB!" W. Shakespeare, *Hamlet*, III, i.

3 - Rocky Mountain Software produces NewWord, a WordStar look-alike whose "Find" command does not have these limitations. See *User's Guide* #9 and the double-issue #11/12 for more information about NewWord. When last tested by the author, NewWord's "Find" command ran so slowly that it seemed less useful than WordStar's, although it was more powerful.

4 - In the section on the "Find and Replace" command, you learn a more efficient way to do this.

5 - WordStar's documentation calls these characters "special characters," but the term "wildcard characters" is accurate and more meaningful.

6 - The ^P is unnecessary in this case, since ^B is not an editing character in a find string. It can't hurt, though.

Z80 Assemblers For 8-Bit CP/M

A roundup of CP/M-80 assemblers: heavy duty programming tools for writing and debugging Z80 assembly language programs.

by Edward J. Joyce

An early version of this review appeared in the now-defunct Microcomputing *magazine (Nov. 1984 issue). Ed Joyce added five more assemblers and revised the entire article for publication in this issue (that means this is version 2.0 of the article).*

This review covers assemblers for the Z80 processor that run under CP/M-80 version 2.2 and produce code for that system. There are other CP/M processor configurations, but the Z80-based CP/M systems rank as the most popular at this time.

Here are two paragraphs introducing assembly language and assembling, which you can skip if you know them.

Assembly language is a powerful tool for writing fast programs, and it usually lets you write programs that occupy less memory than programs written in "high-level" (easy to read) programming languages. Programmers choose assembly language for a variety of different reasons; they might also program in languages like C, Forth, Pascal or Basic. For programs that perform system tasks (reading and writing files on disk, accessing devices through ports, creating screen graphics), programmers usually choose assembly language.

To write a program in assembly language, you type program instructions and comments into a text file, providing enough comments so that you can understand what the assembly language means. When you finish your program, you run another program called an *assembler*, which produces a file of hexadecimal numbers (a ".HEX" file). You then use the LOAD program to translate the ".HEX" file into an executable ".COM" file, which you can run on your Z80-based CP/M computer.

In reviewing assemblers, I found that the basic functions — machine code generation, conditional directive and macro capabilities — differentiate them. Seasoned programmers can fairly gauge an assembler by perusing the user's manual. In most cases, the manual highlights the software's flexibility, power and robustness.

By and large, the single most important attribute is performance: the combination of speed, accuracy and thoroughness that gives you the feeling that the tool does the job. Programmers don't buy assemblers with performance benchmarks that barely edge out an Ada compiler on a Commodore Vic-20.

To separate swift assemblers from sluggish klunkers, I've put eight assemblers through the paces (Sieve benchmarks shown in figures 1 and 2). The individual reviews describe the fine points (and not-so-fine points) of each software package. The accompanying comparison chart summarizes the salient qualities.

MAC: in the Digital Research Tradition

Three years ago I started using Digital Research's MAC as an alternative to ASM, the standard assembler delivered with CP/M. During

cf4
cf39
cf396
cf2

Sahnta Pannutti

that time, the product seems to have changed little, except that it is now shipped with revised, IBM-sized documentation and it's accompanied by several other programs in a package called DR Assembler Plus Tools.

MAC fits the mold of traditional CP/M-80 assemblers. Indeed it traces its lineage to ASM. Macro and cross-reference capabilities set it apart from the older sibling, but there's no mistaking the family resemblance.

Unfortunately, the family resemblance includes several genetic faults. Casually type `MAC↵` on the command line without parameters, for example, and you're appropriately informed that a source file name is required. But before returning to the operating system, MAC sticks two null files named ".HEX" and ".PRN" in the directory, meaningless mementos of the brief visit.

Neither MAC nor ASM support aborting from the keyboard. They don't give up the processor until the job's done, period. Depending on MAC for serious software development means keeping a can of lubricating oil handy for your computer's reset switch.

Massaging Source Code

In another vein, one flaw found in MAC but absent in ASM is uppercase conversion in listings. When generating a listing file, MAC converts all text — data directives, instruction statements and comments — to uppercase regardless of their appearance in the original source code. The statement:

```
ORA A ; clear carry↵
```

Becomes:

```
ORA A ; CLEAR CARRY
```

This unrequested and inexplicable massaging of the original text masks differences between code and comments detracting from the readability of already naturally inscrutable assembly language.

Regardless of its shortcomings, MAC has garnered a solid following over the years. No doubt the popularity in some part stems from its speed. MAC places second in the Sieve benchmarks (figure 1).

A cross reference adds 38 seconds to the benchmark, an inordinately long time explained by the unusual way the cross reference listing is created. The cross reference utility, XREF, reads the listing and symbol files created on disk by MAC. Then it appends the cross reference to a new file with an extension of ".XRF". In the process of generating a cross reference, you end up with two listing files, certainly less than efficient utilization of disk space.

Besides a cross reference, MAC offers two other main functions absent from the standard ASM assembler. It has extensive macro capabilities which are relied on for processing Z80 operation codes (among other things). Normally, MAC recognizes only Intel 8080 mnemonics. A supplied macro library called Z80.LIB handles the instructions unique to the Z80.

Relocatable code is supported through RMAC, a relocatable assembler delivered with the package. Another program in the system, LINK-80, combines RMAC object modules to produce an absolute file.

Dynamic Debuggers

Two of the strongest assets of the package are the dynamic symbolic debuggers, SID and ZSID. SID interfaces with MAC's symbol table to accept symbolic expressions as well as numeric values. SID's utility programs TRACE and HIST aid the debugging process by tracing 256 instructions leading up to a breakpoint and illustrating the relative time spent in selected program segments.

ZSID is the Z80 version of SID. Ironically, it uses Zilog mnemonics instead of Intel mnemonics which means that your MAC source code won't match ZSID's instruction operands.

When I entered the documentation arena, I began to sense that MAC is backed by a major player in microcomputing. About 400 pages in an IBM-sized three ring binder describe the half dozen programs. The indexed manual is typeset and program statements appear in burgundy,

Figure 1. Sieve of Eratosthenes prime number benchmark coded for the Z80 microprocessor in Zilog mnemonics. This program performs 10 iterations of calculating the 1899 prime numbers between three and 16,381.

```
'Generate Sieve of Eratosthenes Primes'                    Z80ASM 1.05 Page   1
SIEVE   Z80

                             TITLE 'Generate Sieve of Eratosthenes Primes'
           0100              ORG    100H

                    ; Equates.
           000D     CR       EQU    0DH      ; carriage return
           000A     LF       EQU    0AH      ; line feed
           0005     BDOS     EQU    0005     ; BDOS entry point

                    ; BDOS Function Calls.
           0009     PSTRING EQU    9        ; print string until '$'

     0100  C3 2171           JP     START    ; skip around data

     0103  0D 0A 0A 31 BGNMSG DB    CR,LF,LF,'10 iterations',CR,LF,LF,'$'

     0117  0D 0A 0A   ENDMSG DB    CR,LF,LF
     011A    0004     PRIMES DS    4
     011E  20 70 72 69       DB    ' primes',CR,LF,LF,'$'

     0129    0040            DS    64       ; reserve 32 stack levels
             0169     STACK  EQU   $
     0169    0002     SAVESP DS    2        ; saves system stack pointer

     016B  00         ITER   DB    0        ; counts iterations
     016C  0000       COUNT  DW    0        ; counts primes
     016E    0002     PINDX  DS    2        ; points to prime multiples
     0170    0002     PRIME  DS    2        ; prime number
     0172    1FFF     FLAGS  DS    8191     ; indicates integers between
                                            ;  1 & 8191 that are prime
                                            ;  0 => not prime
                                            ;  1 => prime
             1FFF     FLAGLEN EQU  $-FLAGS  ; length of FLAGS



     2171  ED 73 0169 START  LD    (SAVESP),SP     ; save system stack pntr
     2175  31 0169           LD    SP,STACK        ; set up local stack
     2178  11 0103           LD    DE,BGNMSG       ; DE -> message
     217B  0E 09             LD    C,PSTRING       ; greet operator
     217D  CD 0005           CALL  BDOS

     2180  21 0172   NXTITER LD    HL,FLAGS        ; initialize FLAGS array
     2183  36 01             LD    (HL),1          ; set first byte of move
     2185  11 0173           LD    DE,FLAGS+1      ; DE -> destination
     2188  01 1FFE           LD    BC,FLAGLEN-1    ; BC <- length of move
     218B  ED B0             LDIR                  ; BC now = 0, use as
```

➤

Figure 1 continued

```
   47                                          ;  FLAGS array index
   48
   49                    ; Check element in FLAGS array.
   50 218D  DD 21 0172  CHKELEM LD     IX,FLAGS      ; IX -) base of FLAGS
   51 2191  DD 09               ADD    IX,BC         ; check element in FLAGS
   52 2193  DD 7E 00            LD     A,(IX+0)      ; A (- FLAGS element
   53 2196  60                  LD     H,B           ; HL (- FLAGS index
   54 2197  69                  LD     L,C
   55 2198  B7                  OR     A             ; prime?
   56 2199  20 11               JR     NZ,FNDPRIM    ;  yes
   57
   58 219B  11 1FFF     NXTELEM LD     DE,FLAGLEN    ; end of array?

'Generate Sieve of Eratosthenes Primes'                 Z80ASM 1.05 Page   2
SIEVE   Z80

   59 219E  ED 52               SBC    HL,DE         ; carry known to be 0
   60 21A0  28 48               JR     Z,ITERDON     ;  yes--next iteration
   61 21A2  79                  LD     A,C           ;  no--increment BC,
   62 21A3  C6 01               ADD    A,1           ;   index to next element
   63 21A5  4F                  LD     C,A
   64 21A6  78                  LD     A,B
   65 21A7  CE 00               ADC    A,0
   66 21A9  47                  LD     B,A
   67 21AA  18 E1               JR     CHKELEM       ; check next element
   68
   69                    ; Found prime.
   70 21AC  E5          FNDPRIM PUSH   HL            ;  yes--save index
   71 21AD  29                  ADD    HL,HL         ; index to multiple
   72 21AE  11 0003             LD     DE,3          ;  PRIME = index+index+3
   73 21B1  ED 5A               ADC    HL,DE
   74 21B3  22 0170             LD     (PRIME),HL    ; save prime
   75 21B6  D1                  POP    DE            ; DE (- index
   76 21B7  19                  ADD    HL,DE         ;  PINDX = index+prime
   77
   78 21B8  22 016E     ZAPMULT LD     (PINDX),HL    ; set multiples non-prime
   79 21BB  B7                  OR     A             ; clear carry
   80 21BC  11 2000             LD     DE,FLAGLEN+1  ; end of array?
   81 21BF  ED 52               SBC    HL,DE
   82 21C1  30 18               JR     NC,INCCOUN    ;  yes
   83 21C3  DD 21 0172          LD     IX,FLAGS      ;  no--indicate non-prime
   84 21C7  ED 5B 016E          LD     DE,(PINDX)
   85 21CB  DD 19               ADD    IX,DE
   86 21CD  DD 36 00 00         LD     (IX+0),0
   87
   88 21D1  ED 5B 0170          LD     DE,(PRIME)    ; point to next multiple
   89 21D5  2A 016E             LD     HL,(PINDX)
   90 21D8  19                  ADD    HL,DE
   91 21D9  18 DD               JR     ZAPMULT
   92
   93 21DB  DD 2A 016C  INCCOUN LD     IX,(COUNT)    ; increment primes found
   94 21DF  DD 23               INC    IX
   95 21E1  DD 22 016C          LD     (COUNT),IX
```

Figure 1 continued

```
21E5  60                     LD    H,B           ; HL <- FLAGS index
21E6  69                     LD    L,C
21E7  B7                     OR    A             ; clear carry
21E8  18 B1                  JR    NXTELEM       ; check next element

                   ; Iteration done.
21EA  3A 016        ITERDON  LD    A,(ITER)      ; last iteration?
21ED  FE 09                  CP    9
21EF  28 0D                  JR    Z,ALLDONE     ;  yes
21F1  3C                     INC   A             ;  no--increment iterations
21F2  32 016B                LD    (ITER),A
21F5  21 0000                LD    HL,0          ; reset count
21F8  22 016C                LD    (COUNT),HL
21FB  C3 2180                JP    NXTITER

                   ; Done--convert result to ASCII via brute force.
21FE  2A 016C       ALLDONE  LD    HL,(COUNT)    ; convert COUNT to 4-digit
2201  DD 21 011A             LD    IX,PRIMES     ;  ASCII decimal
2205  DD 36 00 2F            LD    (IX+0),'0'-1  ; initialize ASCII digit
2209  01 03E8                LD    BC,1000       ; convert thousands
220C  DD 34 00      CLOOP1   INC   (IX+0)        ; simulate divide by 1000

'Generate Sieve of Eratosthenes Primes'                 Z80ASM 1.05 Page   3
SIEVE   Z80

220F  B7                     OR    A             ; clear carry
2210  ED 42                  SBC   HL,BC         ; less than 1000?
2212  30 F8                  JR    NC,CLOOP1     ;  no--keep going
2214  09                     ADD   HL,BC         ;  oops, too much

2215  DD 23                  INC   IX            ; point to second digit
2217  DD 36 00 2F            LD    (IX+0),'0'-1  ; initialize it
221B  01 0064                LD    BC,100        ; simulate divide by 100
221E  DD 34 00      CLOOP2   INC   (IX+0)
2221  B7                     OR    A             ; clear carry
2222  ED 42                  SBC   HL,BC         ; less than 100?
2224  30 F8                  JR    NC,CLOOP2     ;  no--keep going
2226  09                     ADD   HL,BC         ;  oops, too much

2227  DD 23                  INC   IX            ; point to third digit
2229  DD 36 00 2F            LD    (IX+0),'0'-1  ; initialize it
222D  7D                     LD    A,L           ; simulate divide by 10
222E  DD 34 00      CLOOP3   INC   (IX+0)
2231  D6 0A                  SUB   10            ; less than 10?
2233  30 F9                  JR    NC,CLOOP3     ;  no--keep going
2235  C6 3A                  ADD   A,'0'+10      ;  oops, too much

2237  DD 23                  INC   IX            ; point to fourth digit
2239  DD 77 00               LD    (IX+0),A      ; save least sig. digit

223C  11 0117                LD    DE,ENDMSG     ; DE -> message
223F  0E 09                  LD    C,PSTRING     ; display results
```

Figure 1 cont. on pg. 40

Overview of the Comparison Chart

Many of the chart categories are self-explanatory. Others warrant explanation. An assembler version of the gold standard of benchmarks, the Sieve of Eratosthenes prime number generator, was chosen to peg performance. This program, shown in figure 1, performs 10 iterations of calculating the 1899 prime numbers between three and 16,381.

Three variations of the benchmark were executed: assembling the Sieve program without a listing; assembling with a listing created on disk; and assembling with a listing and cross-reference created on disk. The benchmarks reflect the time required to assemble the source code and produce a command file (".COM" file). These times include the overhead in loading the assembler and, if necessary, related programs such as a linker and/or CP/M's LOAD utility.

The tests were conducted on a Xerox 820 system with two single sided, single density eight-inch disks and 64K of RAM. The Xerox 820 runs its Z80 microprocessor at 2.5 MHz. The Sieve program, incidentally, executes in 17 seconds on the Xerox. Compare this to 35 seconds for Pascal MT/+ and 2380 seconds for Microsoft Basic on the same machine and you begin to understand why assembler is still the language of choice among programmers who handcraft code for ultimate speed.

Some of the benchmarks in the chart may fall short of advertised claims of a product's performance. This doesn't necessarily mean that the advertisements or the benchmarks are wrong. More than likely, the discrepancy can be accounted for by the nature of the tests chosen to demonstrate performance. The benchmark chosen here denotes assembly speed for the Sieve program, a relatively short program of 148 source code lines. It may or may not showcase a product at optimal performance. Some assemblers, for example, may shine while handling large programs that require constant paging of symbol table and intermediate files from disk. Other programs might work efficiently at linking dozens of modules. At any rate, the Sieve benchmark provides only a relative measure of assembling a single program.

After performance, the comparison chart lists the presence or absence of noteworthy characteristics. The 'generate ".COM" or ".HEX" file' category refers to the output of the assembler. Some programs produce an executable ".COM" file. Others produce hexadecimal format with a file extension of ".HEX" that must be converted to a ".COM" file by the CP/M LOAD utility program. Still others produce either type of file. For the CP/M environment, producing a ".COM" file is more convenient and much preferred.

The mnemonic type classifies the instruction set used by the assembler in the *standard* configuration. The original Digital Research ASM assembler supplied with CP/M uses Intel mnemonics. Most Z80 assemblers tend to rely on Zilog mnemonics. The difference between the two is just enough to perpetually stymie the novice and occasionally trick the veteran.

Software shops that already have a significant investment in programs coded in a particular style of mnemonics should be especially concerned with this category. The benefits of a new assembler may be wiped out by the hassle of dealing with an alien set of mnemonics.

Abort capability on the comparison chart simply indicates whether or not you can terminate or interrupt an assembly from the keyboard. Surprisingly, many programs plod along in their own little world, oblivious to all but the reset button or power switch.

The final two categories, diagnostic error messages and overall rating, were labeled as good, fair or poor. The quality of diagnostic messages can make or break an assembler. On one extreme, some programs indicate a syntax error with an arbitrary number or single character code which needs to be deciphered in a manual. Other programs provide a textual description which pinpoints the problem much quicker. The error messages were monitored while the assembler tangled with a source code file containing a dozen errors of various types.

Overall rating relates to the general feel of the product. It conveys the flavor of the meal after you know the ingredients, calories and cooking time. This category not only compares the product with the other assemblers but it also pits the program against popular commercial software.

Unfortunately, none of the overall ratings warrant an excellent designation. Microcomputer assemblers, the most essential tools for software development, still fall dismally short of the high standards established for user friendliness and functionality now common place in the applications software arena.

Chart follows on next two pages.

STONEVILLE ACCOUNTING
THERE MUST BE A BETTER WAY!
BILLS
BILLS
BILLS
BILLS
BILLS

In The Beginning...

There was the "Shoe-Box" accounting system...

But Now There's Mini-Ledger

It doesn't make sense to try to tell you about **Mini-Ledger** in an ad this size. That is especially true since the arrival of **Accounts Payable** and **Accounts Receivable.**

So, if you want to learn about **Mini-Ledger** accounting, **order our interactive brochure disk.** The disk will not only show you what the programs do, but how easy they are to use.

The brochure disk is **$5.00.*** At the very least you will get a disk, and who knows, you might even discover why you bought your computer in the first place.

*credited toward purchase

With Mini-Ledger, those prehistoric days of Shoe-Box accounting are over.

PARADIGM CONSULTANTS

MINI-LEDGER

39243 Liberty Street, Suite L
Fremont, CA 94538 **(415) 796-0543**
Outside California Call **(800) 821-8407**

CP/M is a registered trademark of Digital Research Corporation

Circle number 12 on the reader service card.

Comparing Z80 Assemblers

		Time to produce ".COM" file					
Name/Manufacturer	Price	Sieve with no listing	Sieve with listing	Sieve with listing & cross-reference	SUBMIT (batch mode) compatible	Generate ".COM" or ".HEX" file	Mnemonic Type
MAC version 2.0 Digital Research Inc. P.O. Box 579 Pacific Grove, CA 93950 800-227-1617, 408-649-3896	$200.00	53 seconds	61 seconds	99 seconds	Yes	HEX	Intel
Z80ASM version 1.05 SLR Systems 1622 N. Main St. Butler, PA 16001 800-833-3061, 412-282-0864	$199.95	23 seconds	35 seconds	35 seconds	Yes	either	Zilog
ASMZ80 version 3.6C Relational Memory Systems (RELMS) 1650-B Berryessa Rd. San Jose, CA 95133 800-448-4880, 408-729-3011	$395	62 seconds	73 seconds	87 seconds	Yes	*	Zilog
Omnicoder version 1.0 Mockingbird Data Systems 2296 Hoover Rd. Grove City, OH 43123 614-875-0753	$90	108 seconds	135 seconds	not available	Yes	COM	Zilog
SMAL/80 version z4.3 Chromod Associates 1030 Park Ave. Hoboken, NJ 07030 201-653-7615	$149.95	68 seconds	116 seconds	not available	Yes	either	Hybrid
Macro-80 version 3.44 Microsoft Corp. 10700 Northrup Way Bellevue, WA 98004 (800) 426-9400, (206) 828-8080	$200.00	46 seconds	56 seconds	84 seconds	Yes	**	Zilog & Intel
XASMZ80 version 1.04 Avocet Systems P.O. Box 490 Rockport, ME 04856 (800) 448-8500, (207) 236-9055	$250.00	30 seconds	37 seconds	not available	Yes	HEX	Zilog
ASM version 3.01a 2500AD Software Inc. P.O. Box 4957 Englewood, CO 80155 (303) 790-2588	$99.50	127 seconds	144 seconds	161 seconds	Yes	**	Zilog ***

* ASMZ80 produces intermediate code that is converted by a utility program into a ".HEX" file.

Abort Capability	Interactive Mode	Significant Characters in Labels	Macros	Conditional Assembly	Linker	Relocatable Code	Diagnostic Error Messages	Overall Rating
No	No	16	Yes	Yes	Yes	Yes	Poor	Fair
Yes	Yes	16	Yes	Yes	Yes	Yes	Good	Good
No	No	6	Yes	Yes	Yes	Yes	Poor	Poor
Yes	Yes	8	Yes	No	Yes	Yes	Poor	Fair
No	No	6	Yes	Yes	No	Yes	Fair	Fair
No	Yes	16	Yes	Yes	Yes	Yes	Poor	Fair
Yes	No	8	No	Yes	No	No	Poor	Poor
Yes	Yes	10	Yes	Yes	Yes	Yes	Good	Good

** Produces object file that goes to linker for ".COM" or ".HEX".

*** Has Intel to Zilog converter, too.

Figure 1 continued

```
   144 2241  CD 0005            CALL    BDOS
   145
   146 2244  ED 7B 0169         LD      SP,(SAVESP)      ; restore system stack pntr
   147 2248  C9                 RET                      ; return to CP/M
   148                          END
  0 Error(s) Detected.
  8521 Absolute Bytes. 27 Symbols Detected. ¶ 'Generate Sieve of Eratosthenes Primes'       Z80ASM 1.05 Page   4
Symbol Table:

21FE  ALLDONE          0005  BDOS             0103  BGNMSG
218D  CHKELEM          220C  CLOOP1           221E  CLOOP2
222E  CLOOP3           016C  COUNT            000D  CR
0117  ENDMSG           1FFF  FLAGLEN          0172  FLAGS
21AC  FNDPRIM          21DB  INCCOUN          016B  ITER
21EA  ITERDON          000A  LF               219B  NXTELEM
2180  NXTITER          016E  PINDX            0170  PRIME
011A  PRIMES           0009  PSTRING          0169  SAVESP
0169  STACK            2171  START            21B8  ZAPMULT

'Generate Sieve of Eratosthenes Primes'                Z80ASM 1.05 Page   5
Cross Reference:

21FE  ALLDONE       104   112
0005  BDOS            7    40   144
0103  BGNMSG         14    38
218D  CHKELEM        50    67
220C  CLOOP1        116   119
221E  CLOOP2        125   128
222E  CLOOP3        134   136
016C  COUNT          25    93    95   108   112
000D  CR              5    14    15    16    19
0117  ENDMSG         16   142
1FFF  FLAGLEN        32    45    58    80
0172  FLAGS          28    32    42    44    50    83
21AC  FNDPRIM        56    70
21DB  INCCOUN        82    93
016B  ITER           24   102   106
21EA  ITERDON        60   102
000A  LF              6    14    14    15    15    16    16    19    19
219B  NXTELEM        58    99
2180  NXTITER        42   109
016E  PINDX          26    78    84    89
0170  PRIME          27    74    88
011A  PRIMES         17   113
0009  PSTRING        10    39   143
0169  SAVESP         22    36   146
0169  STACK          21    37
2171  START          12    36
21B8  ZAPMULT        78    91
```

END FIGURE 1

setting them off from the black text of the rest of the manual. A pocket guide keeps operation instructions for ZSID at your fingertips. Based on documentation alone, MAC leaves other products far behind.

In the final analysis, though, I'd rather have a good program with mediocre documentation than a mediocre program with good documentation (provided, of course, I can't get a *good* program with *good* documentation).

SLR's Z80ASM assembler stands head and shoulders above most other assemblers. If I had to summarize the product in two words, they'd be speed and flexibility.

MAC resembles a Model-T software product. Sure, it's transportation, but not the ultimate in speed and convenience. Since Digital Research started pursuing the 16-bit market they seem satisfied with the status quo in their 8-bit products. Meanwhile, a host of other vendors have surfaced providing state of the art development tools for CP/M-80.

To sum up, the investment in the DR Assembler Plus Tools may be worthwhile for ZSID, but if you're looking primarily for an assembler then save your shekels. Other products have this package beat.

Z80ASM From SLR Systems: Fast and Flexible

Initially, I didn't expect much from SLR Systems' Z80ASM. My skepticism started before I got past page one of the user's manual. The opening page outlines a command line format that follows the clumsy precedent established by Digital Research's ASM assembler. Namely, the drive parameters for the source, object and listing files appear where you expect the file extension to be.

For example, to assemble the source file TEST.Z80 on drive A and direct the object and listing files to drive B, you type:

```
A>Z80ASM TEST.ABB↵
```

The extension of the file is always assumed to be ".Z80" (TEST.Z80).

Beyond the command line format, SLR's assembler fortunately shares little else with its Digital Research counterpart. Overall, it stands head and shoulders above ASM and most other assemblers. If I had to summarize the product in two words, they'd be speed and flexibility.

Configuration Options

The flexibility asset was revealed when I configured the software. The configuration program controls the settings of over 25 default parameters. These range from inclusion of time and date on listings (for CP/M Plus environments) to extensions of input and output files to page width and length.

My two favorite configuration options are error limit and error paging. Error limit sets a cut-off point for a maximum number of errors. If that limit is reached, the assembly is terminated. This option comes in handy for controlling the domino effect which occurs when one error causes a cascade of errors, such as one missing macro library include statement generating a stream of undefined operation code errors.

The error paging option specifies a maximum count of errors displayed on the console screen before pausing for operator intervention. Displaying error messages on the console screen is of little value if they flash by at a baud rate approaching the speed of light. This count stops the display in its tracks and waits for you to signal continuation. While assembling a program you no longer have to hover over the keyboard, left hand poised on Control-S, to avoid missing critical messages.

Once you've tailored the assembler to your personal *modus operandi*, you're ready for a test drive. Before you take off, I suggest buckling your seatbelt. This program doesn't bill itself as "super fast" for nothing!

The benchmarks in table 1 support the speed contention. Z80ASM assembles the Sieve program shown in figure 1 in less than half the time of the other assemblers. Part of the turbo-performance stems from the fact that the load step is eliminated completely. Programs such as Digital Research's ASM and MAC generate ".HEX" files which must be subsequently fed into the LOAD utility to produce ".COM" files. Z80ASM directly generates the ".COM" file. This shortcut saves 11 seconds, the time required to execute LOAD, in the Sieve benchmark. (You can direct Z80ASM to generate a ".HEX" file, if you want.)

Interactive Mode

Z80ASM cuts assembly times further when you process multiple files. The program offers an interactive mode analogous to the PIP utility program. You can assemble a batch of files without re-invoking the program. Multiple assemblies can also be specified on the command line. For instance, the following command line assembles three files in one fell swoop.

```
A>Z80ASM SIEVE,DUMP,BIOS↵
```

Several of the product's less prominent features are revealed when you examine a program listing. Referring to figure 1, which was created by Z80ASM, you can see that both upper and lower case characters are supported. This is a refreshing change from the assemblers that insist on converting everything to uppercase, blurring the distinction between instructions and comments.

The values of equates (EQU) and define space (DS) directives are indented slightly to set them apart from data and instruction codes. Software jocks probably won't invest in the package based on this point alone, but it exhibits the attention to detail which guided the system's developers.

Notice also that 16-bit values are printed in logical order. The code for statement 46, the `CALL BDOS` instruction, appears as `CD 0005` instead of the usual `CD 0500`. During configuration you have the choice of ordering these values either way. The former makes it easier to read — you don't have to mentally flip the bytes to identify an address. But the latter reflects the actual image in memory. In any case, flexibility reigns again, giving you the choice.

When forming instruction operands and expressions you have a rich assortment of 22 operators to chose from. These include division, multiplication, shifting, and logical *and*, *or* and *exclusive or*.

The listing in figure 1 concludes with a routine cross reference. For the Sieve program, generating a cross reference adds *no* discernible overhead to the benchmark, a feat matching the often-promised but rarely-delivered free lunch. My only suggestion would be to distinguish label references from label definitions by posting an asterisk next to the statement numbers of the definitions.

On the whole, Z80ASM lives up to its self-appointed sobriquet of a "super fast development package for software entrepreneurs." To support my observations, I called Ed Mitchel, a programmer at Industrial Computer Equipment Corporation. "We have been using Z80ASM since September of 1983," Mitchel explained. "At the time I was looking for a mainframe-caliber development tool and I almost gave up in disgust. Then I chanced upon SLR's program."

"The system cut 20 minute assemblies to six minutes and in a one-year project paid for itself 100 times over."

Do you have any complaints? "Yes," Mitchel replied. "Only that it wasn't available three years ago."

Need I say more?

ASMZ80 From RELMS: Where's the Beef?

Reviewing the RELMS ASMZ80 assembler, like the SLR system, was another case of being betrayed by initial impressions, but of a different sort. For $395, I expected a sophisticated development tool, maybe a micro version of the IBM Waterloo mainframe assembler. Alas, the sophistication of ASMZ80 ends with the price.

My disappointment began when I unwrapped the documentation from RELMS (Relational Memory Systems). The 100 plus pages were delivered loose leaf form sans binder. The descriptions contained within of the assembler, linker and relocator rate no higher marks than the physical packaging.

Parts of the user's manual break my reading rhythm, such as the reference to "reversed" words meaning ostensibly reserved words. Other sections are flat-out wrong. The license agreement, for example, spells out a procedure for backing up the distribution disk using a supplied program called DGEN. DGEN was nowhere to be found, and, as the vendor later admitted, doesn't even apply to the CP/M version of the assembler.

The description of the title directive says that title headings may be up to 50 characters. The program acted to the contrary, truncating my titles at 40 characters.

Fallacies

Another fallacy was the sentence "A four-byte patch area is generated when encountering a blank line or assembler directive." I reread that statement several times and wondered if such a patch area would be classified as a feature or bug. Then I considered removing the generous sprinkling of blank lines I had in the Sieve source code for delineating different logical parts of the program. But a quick experiment showed that blank lines and directives generated no code, so I annotated that clause as another case of the manual missing its mark.

When the manual does tell you something correct, it resorts to overkill to the third degree. Two nearly identical back-to-back sections describe the command line format. I was forced to compare the page numbers to see if a set of pages had been duplicated inadvertently. But the page numbers checked out — I suppose two sets of redundant instructions are better than none.

By this point in the evaluation, I was convinced that my four C-note investment didn't go for a Pulitzer Prize-winning user's manual, so I looked to the software for comfort. I invoked the program to check for an interactive mode. ASMZ80 responded with the very informative message, `ERROR 0023 USER PC 047B9H`. A little leafing through the manual lead to a translation in Appendix F citing error 23 as "no file name specified for a disk file," in other words you must specify the input file name on the command line. No explanation was offered for the `USER PC` phrase, something I figured was meant for an elite, handpicked audience like the encrypted expiration codes found on cereal boxes in the supermarket.

The curt, coded flavor pervades ASMZ80's diagnostic messages. The messages are meaningless without a manual propped next to the console screen. In a few cases, they're actually misleading. For instance, neglecting to terminate a program with an END statement causes ASMZ80 to issue an

Z80ASM offers an interactive mode that lets you assemble a batch of files without re-invoking the program. Multiple assemblies can also be specified on the command line.

`F` error meaning an invalid comment field.

Abridged error messages could be overlooked if the assembler delivered superior performance, but the benchmark timings are nothing to write home about. Furthermore, ASMZ80 complicates, rather than simplifies, the process of creating a ".COM" file. The assembler produces an Intel object file. That file must then be fed into a RELMS utility program which converts it into Intel HEX format. Finally, the CP/M LOAD utility converts the HEX format into a ".COM" file.

Other liabilities include a six character label restriction, 30 macro limitation and lack of a keyboard abort. Six character label limitations have no place in the programming tools of the 1980's. Symbols become so abbreviated, you might as well use the hexadecimal address.

Constraining macros to a maximum of 30 definitions seems to be another arbitrary limitation. Certainly, you won't be able to rely on ASMZ80 with only 30 macros for those situations when you want to define an alternate instruction set with macros in order to cross-assemble code for a non-Z80 processor.

Cardinal Sin

And lack of a keyboard abort — what can I say about that cardinal sin? Would you buy an oven if you couldn't shut it off until it cooked for the designated time? Or a car that didn't stop until it reached the destination even if you forgot to bring your wallet? I'm sorry, but programs that won't spend a piddling few milliseconds to occasionally check the keyboard should be banned by federal edict.

Given its flaws, I can't recommend RELMS' Z80 assembler for serious software development. Still it has some redeeming points which should be mentioned for the record.

Twenty operators are available for constructing expressions of virtually any conceivable form. ASMZ80 offers the garden variety arithmetic (`+`, `-`, `*`, `/`) and logical (`not`, `and`, `or`) operators. They then supplement these with the modulo or remainder function and, of all things, exponentiation. That's the first time I've come across an assembler, or any compiler for that matter, that supports exponentiation in constant expressions.

Staff personnel at RELMS support the product with a toll-free 800 hotline connected to their Silicon Valley office. They also offer a free, one year update for new releases of the software.

Besides the ASMZ80 assembler, RELMS's catalog is chock-full of hardware and software products for the software development environment. The stable includes Iceboxes (In-circuit emulators) with over 256K of ROM and RAM, communications programs and CP/M-based cross-assemblers which generate code for 17 different microprocessors.

I suspect that the diversity of the assembler product line accounts for some of the flaws found in ASMZ80. The same core program is probably used in all of the assemblers. Modules or tables are then replaced to generate, say, a 6800 instead of a Z80 assembler. While a universal assembler that conveniently generates code for a variety of machines is a commendable endeavor, it inevitably sacrifices the optimal performance achieved by the fine-tuned single-system assemblers.

I could live with the shortcomings of the RELMS assembler, if it weren't for the price. But after grappling with the problems and realizing that I paid close to four hundred dollars for the opportunity, I'm left with only one conclusion. Where's the beef?

Mockingbird's Omnicoder: Newcomer with Promise

Omnicoder's most notable characteristic invites both blessing and curse. The system came to life recently. Obviously, entrepreneurs at Mockingbird Data Systems have no fifth generation lineage or a large customer base to refer to in enticing advertisements. On the other hand, author Jerry Mercier has had the opportunity to learn from the mistakes and shortcomings of competitive products. He's in the position of a Monday morning quarterback plotting strategy for yesterday's game.

Mercier scored instant kudos with this reviewer by incorporating help screens in the system's assembler and link editor. The programs work in batch and interactive modes and in the latter case general help information is available at the touch of a key.

Innovative Cross Reference

At this time, the system lacks a conventional cross reference, but a margin reference provides similar information. In the left hand column between the assembled code and source statement, Omnicoder lists the line number defining the operands referred to in the statement. Although I've used this feature on high-level language compilers, Omnicoder marks the first time I've seen it in an assembler. Couple the margin reference with the general cross reference planned for the system (existing users can update at no charge) and you'll have one of the best cross references available for debugging.

For mnemonics, Omnicoder uses Zilog format with the exception of the assembler directives which fall into a hybrid class. For example, the following Zilog define byte statement:

```
DB  'primes',0DH,0AH,0AH,'$'
```

— translates to the following Omnicoder statement:

```
%DATA S('primes'), S(X'0D0A0A'),
S('$')
```

This form, in the words of the user's manual, "is designed to avoid conflict with machine instructions of various CPU's." In the future, the system is expected to support multiple processors.

Noble Goal, But ...

Personally, I regard yet another flavor of Zilog's *de facto* standard as a nuisance. Supporting multiple processors is a noble goal, but fine-tuning for a particular machine and, in this case, clarity and free format inevitably seem to be sacrificed.

Another liability on Omnicoder's ledger is performance. The Sieve assembly without a listing ticks away for 73 seconds. Throw in another 35 seconds for loading and executing the mandatory linker and you're up to a hefty 108 seconds. No speed records set there.

I suspect that two factors contribute to the unremarkable showing. First, all programs must be linked, even if they consist of only one absolute object file. This step alone adds more time than some systems require to assemble and produce the entire command file. Second, given the relative infancy of the program, it probably has yet to benefit from optimization.

Omnicoder's immaturity shows up in other operational areas, too. The program offers an abort exit, but when you take advantage it, work files with zero allocation are left on the disk (shades of MAC).

Selecting the option to create a listing file on disk suppresses display of error messages on the screen. You're forced to wait for the completion of the assembly to review error messages.

Discounting the imperfections, Omnicoder has the potential to establish new horizons in user friendliness for the assembler programmer. The advent of help screens in a machine code assembler proves that besides users, programmers also need to be coddled.

Mockingbird Data Systems also hopes to set precedents in other areas, mainly cost effectiveness. The $90 price tag wins a prize for the least expensive of the eight assemblers reviewed. Additionally, the generous "site" licensing plan contributes to the economy of the investment. According to product literature, "a site is loosely defined as that portion of a single building occupied by the licensee; reasonable exceptions will be considered upon written request."

Omnicoder should eventually mature into a heavy duty software development tool. It's still several notches below the thoroughbreds, but give it a release or two and the gap may shrink to a photo finish.

For $395, I expected a sophisticated development tool, maybe a micro version of the IBM Waterloo mainframe assembler. Alas, the sophistication of ASMZ80 ends with the price.

Chromod's SMAL/80: Based on Bell Lab's Structured Assembler

If the developers of SMAL/80 at Chromod Associates have their way, assembly language programming will soon be a relic of the past. Their system purportedly streamlines assembler coding by masking it as a high-level language.

How is that accomplished? I'll let the code speak for itself. Figure 2 shows the SMAL/80 version of the Sieve benchmark. While the SMAL/80 mnemonics bear little resemblence to conventional assembler, they should ring a bell with Modula-2, Pascal and C programmers.

SMAL/80 (Structured Macro-Assembly Language) traces its roots to Bell Labs. There, in 1974, Charles Popper developed an 8080 language and described it in the IEEE Computer Society Fall conference proceedings.

Popper's design stemmed from his dissatisfaction with Intel's 8080 mnemonics. In his opinion, they only vaguely matched the instruction's operation, a belief reinforced by the symbolic high-level notation microprocessor manufacturers used in their publications to explain what the mnemonics meant. So motivated, Popper concocted a symbolic notation that included control structures commonly found in Algol and C.

SMAL/80 automatically generates code for BEGIN/END, IF/THEN/ELSE and LOOP/REPEAT constructs. It also adds indentation which illustrates the logical structure of an algorithm.

A macro preprocessor optionally allows the standard set of symbols to be substituted with programmer-defined identifiers. With this capability, you can replace the assignment operator = with := or <-. The preprocessor handles more exotic changes, too, enabling program key words to be written in another tongue, say French.

My journey into the land of SMAL/80 started with a translator utility program which, in the words of the documentation, "automatically upgrades assembly code to SMAL/80."

Longest SUBMIT File Ever

The translator program works from within a gargantuan nested SUBMIT file, very likely a candidate for inclusion in the *Guinness World Book of Records* as the longest file to ever çarry a ".SUB" extension. I worked for over two hours trying to translate a 150-line program of Zilog mnemonics and finally surrendered when an ancillary utility program hung the system. From that point, I resorted to good old manual editing.

Certainly the documentation contributes to part of the problem. At first glance, it seems more than adequate. Over 100 pages describe the language, compiler (SMAL/80), linker and instruction tables. The instruction tables show equivalent instructions in SMAL/80, Intel and Zilog mnemonics.

Two other manuals round out the

printed fare. A pocket guide condenses much of the reference material in a convenient format. And a book, *Structured Microprocessor Programming*, an optional item priced at $20, introduces the novice to microprocessor machine language coding in the style of SMAL/80.

Despite this volume of printed information, critical details fall through the cracks. The procedure for producing a ".HEX" file, the key step preceding generation of a ".COM" file, is relegated to a footnote appended to a table describing assembler directives.

Some Z80 instructions, such as the following, are completely missing from the instruction tables:

```
LD (IX+d),n
```

(Load operand `n` to the address specified by the sum of the index register IX and the displacement `d`)

Overall, the documentation seems to be strong on SMAL/80, but weak in CP/M implementation details.

Addiction

Once you've grasped the system's operation, writing programs is relatively straightforward since the instructions truly resemble a high-level language. The mnemonic `A=D` is vastly simpler and clearer than the Intel form of `MOV A,D` or the Zilog equivalent, `LD A,D`. Undoubtedly, SMAL/80's symbolic notation can be addicting.

While these mnemonics logically correspond with the operation of the instruction, they don't mesh nearly as neatly with debuggers like DDT or ZSID. You have to mentally convert SMAL/80 to Intel or Zilog to trace a program from a listing. This drawback should also be weighed if you depend on manufacturer manuals for instruction specifications.

On the performance end, the system chalks up a respectable benchmark of 68 seconds, which lends credence to Chromod Associates' claim of assembler-like execution speed. Since the package lacks a cross-reference generator, that benchmark couldn't be timed.

If you miss the high-level constructs of languages like Pascal, then SMAL/80 may be just your cup of tea. Chromod Associates sells it on a try-it-you'll-like-it no risk basis. They'll unconditionally refund the $149.95 price within 30 days of purchase.

I still cast my lot with assemblers that use conventional mnemonics. The mnemonics of Zilog and Intel seem to have been irrevocably etched on my gray matter. At any rate, if you haven't been so tainted, scope out SMAL/80. It embodies a hard to beat combination of high-level-language convenience with machine level performance.

Microsoft's M80: Patriarch of the Pastures

Microsoft's Macro Assembler product has undergone a transformation. Collectively packaged as the M80 assembler, L80 linker, CREF80 cross-reference generator, and LIB library manager, the Macro Assembler has trod the pastures of computing for at least six years. Today, it sports a new hide, a nifty skin of revised documentation. The physical package sparkles so much that you'd swear it's sugar coated. But underneath the surface lives the same old beast and after eons of active duty — in terms of software life cycles — the animal is beginning to show its age.

Technical Writing Kudos

Before dissecting the anatomy, let's point out the positive aspects of the facelift. First and foremost, Microsoft's technical writing department deserves kudos. In general, the IBM-sized manual reflects the professional attention found in the documentation of Multiplan and other Microsoft products. The binder is packed in the transparent plastic EaselBox, characteristic of the Bellevue, Washington company. The EaselBox serves as both bookstand and sturdy storage case for manual and single distribution diskette.

A green and beige color scheme coupled with bold, typeset text painlessly conveys the printed page to the reader's optic center. Thumb tabs, appendixes and an index facilitate quick reference. The appendixes that list Z80 and 8080 opcodes (in Zilog and Intel formats) are particularly handy. The assembler manuals of many other vendors lack even a cursory list of these opcodes. In those cases, the forlorn programmer is forced to leaf through Zilog or Intel product specifications or assembler tutorial books, which may or may not be available during the heat of the battle.

Regarding content, the documentation does a reasonable job, but it can only do so much when the software embodies poor design. To specify the command line, for example, the earnest user must wrestle with a one-of-a-kind, esoteric format. Namely, the command `M80 TEST.REL,TEST.PRN = TEST.ASM/L` ↵ assembles program TEST.ASM and routes the object code to TEST.REL and the listing file to TEST.PRN. The genealogy that led to this protocol would undoubtedly stifle the most ardent students of *Trivial Pursuit*, computer version.

Unpredictable Command Line

Even the author of the reference manual had trouble figuring out the command line. While explaining assembler options through the example, `M80 NEIL, = NEIL` ↵, the manual states, "MS-MACRO [M80] will *probably* ignore the comma and assemble the source file into a REL file. *It is possible* that MS-MACRO *might* return a `COMMAND Error` message [emphasis added]." Perhaps the unpredictable behavior of the command line parser is tied to a Zodiac sign or the Dow Jones Industrial Average.

With the exception of the homebrew command line, M80 virtually mirrors Digital Research's assembler, MAC. Among their numerous similarities: they both retail for $200, include documentation on IBM-sized manuals and generate cross-references by producing new print files through utility programs external to the assembler. In performance, M80 does nose out the offering of its chief competitor. Microsoft's product registers 46 seconds for the Sieve assembly benchmark, about 13 percent less than MAC's timing.

Another dissimilarity lies with the method for switching between 8080 and Z80 mnemonics. Digital Research's MAC essentially uses Intel 8080 mnemonics. Z80 instructions are contained in a macro file called Z80.LIB. Microsoft's M80, on the other hand, allows the programmer to choose between Intel 8080 or Zilog Z80 mnemonics by including a parameter on the command line or by inserting an assembler directive in the source code. The directive `.8080` selects Intel mnemonics and `.Z80` selects Zilog. M80's method stands as a head and shoulders improvement over a macro file.

One of the most attractive features of Microsoft's macro assembler package is its direct compatibility with high level languages. Through the linker, L80, a programmer may combine software developed in Microsoft's assembler, COBOL, FORTRAN and Basic compilers. Actually, any language compiler that produces object code in the specified format (spelled out in the documentation) will mesh with the linker.

On the whole, this product earns a lackluster fair rating. The documentation wins honorable mention and the benchmarks clock in at above average, but without question the software still languishes far behind state-of-the-art competitors in versatility and performance. If you're planning to mesh code with programs developed in high-level languages, you might make a case for including this package in your programming tool chest. The more innovative assemblers, however, also have options for producing Microsoft-linker compatible files. That leaves Microsoft's Macro Assembler without much to offer, except mediocrity . . . and old age.

Avocet: Choice of Professionals?

I've always been curious about the Avocet Systems assembler. For the past four years, the Rockport, Maine, company (based formerly in Dover, Delaware) has touted it as "the choice of professionals" in advertisements gracing the pages of *Byte* magazine. Based on longevity alone, the Avocet assembler would share top honors with Digital Research's MAC and Microsoft's M80.

Recently, I finally had the opportunity to plug this 13K program into the circuits of my Z80 for a firsthand look. According to Avocet's marketing entrepreneurs, I would be joining NASA, IBM, Xerox and "hundreds of other organizations" that use Avocet assemblers. In retrospect, I can understand how the performance of this assembler has enamored a significant customer base. On the other hand, it lacks several essential ingredients that one expects from a professional software development tool.

Avocet Product Line

Before delving into the details, an examination of the Avocet product line will lend some perspective to this review. The company sells cross-assemblers for just about any breed of microprocessor. The assemblers run on CP/M-80, CP/M-86 and MS-DOS systems. Target microprocessors span a wide list encompassing 16 machines. These range from the familiar — Z80, 6502 and 68000 — to the obscure — 1802, 8051 and COPS400.

Each of these cross-assemblers operates through a common underlying syntax. This similarity suggests that the assembler is table driven. Different tables of operands and opcodes would be used to produce the various versions of the assembler which are sold.

Out of the eight software vendors reviewed in this article, Relms and 2500AD offer a similar assortment of assemblers. For that reason, I initially expected Avocet's assembler to post average or below average performances like its Relms and 2500AD counterparts. To my surprise, it ranked second in the benchmark timings.

SMAL/80 embodies a hard to beat combination of high-level-language convenience with machine level performance. Undoubtedly, SMAL/80's symbolic notation can be addicting.

Avocet's assembler finished the Sieve benchmark in 30 seconds, seven seconds shy of the fastest timing of SLR Systems and a full 16 seconds quicker than Microsoft's M80. Clearly, this program was not merely a warmed over conversion from another system. It operated efficiently in the CP/M environment.

Shortcuts

Part of the explanation for the racy performance could lie with implementation shortcuts. Labels are limited to eight characters, nothing to write home about there. Listings do not include statement line numbers. Worse yet, there is no provision for macros. That handicap in itself will probably keep many programmers away from the Avocet fold.

Another glaring problem is the absence of a cross-reference generator. Only two other assemblers in the eight reviewed in this roundup shared that disadvantage. But prices that undercut Avocet's by at least one hundred dol-

Figure 2. Sieve of Eratosthenes prime number benchmark coded for the Z80 microprocessor in SMAL/80, Structured Macro-Assembly Language.

```
Version z4.3, COPYRIGHT(C) 1984 by ChRoMoDassociates

ChRoMoDassociates SMAL COMPILER, Vz4.3,          PAGE 0001
Sieve of Eratosthenes--SMAL/80

INTERNAL SYMBOLS
EA BDOS     X'0005'  LA BGNMSG   X'0103'  LA CLOOP1   X'2207'
LA CLOOP2   X'2219'  LA CLOOP3   X'2229'  LA COUNT    X'016C'
EA CR       X'000D'  LA ENDMSG   X'0117'  EA FLAGLEN  X'1FFF'
LA FLAGS    X'0172'  LA INCCOUN  X'21D1'  LA ITER     X'016B'
EA LF       X'000A'  LA PINDX    X'016E'  LA PRIME    X'0170'
LA PRIMES   X'011A'  EA PSTRING  X'0009'  LA SAVESP   X'0169'
EA STACK    X'0169'  LA START    X'2171'  LA ZAPMULT  X'21AE'

EXTERNAL SYMBOLS


ChRoMoDassociates SMAL COMPILER, Vz4.3,          PAGE 0002
Sieve of Eratosthenes--SMAL/80

                #0001:          TITLE 'Sieve of Eratosthenes--SMAL/80'
                #0002:          MACHINE = Z80;
                #0003:          HEX = 16;
0000 A          #0004:          ORIGIN  X'0100';
0100 A          #0004:
                #0005: /* Equates. */
000D A          #0006: CR       EQU     X'0D'   /* carriage return  */;
000A A          #0007: LF       EQU     X'0A'   /* line feed */;
0005 A          #0008: BDOS     EQU     0005    /* BDOS entry point */;
                #0009:
                #0010: /* BDOS Function Calls. */
0009 A          #0011: PSTRING EQU      9       /* print string until '$' */;
                #0012:
0100 A C37121   #0013:          GOTO    START   /* skip around data */;
                #0014:
0103 A          #0015: BGNMSG: BYTE     CR,LF,LF,'10 iterations',CR,LF,LF,'$';
0103 A 0D       #0015:
0104 A 0A       #0015:
0105 A 0A       #0015:
0106 A 31302069 #0015:
010A A 74657261 #0015:
010E A 74696F6E #0015:
0112 A 73       #0015:
0113 A 0D       #0015:
0114 A 0A       #0015:
0115 A 0A       #0015:
0116 A 24       #0015:
0117 A          #0016: ENDMSG: BYTE     CR,LF,LF;
0117 A 0D       #0016:
0118 A 0A       #0016:
```

➤

Figure 2 continued

```
0119 A 0A        #0016:
011A A           #0017: PRIMES: RESERVE 4;
011E A           #0017:
011E A 20707269  #0018:         BYTE    ' primes',CR,LF,LF,'$';
0122 A 6D6573    #0018:
0125 A 0D        #0018:
0126 A 0A        #0018:
0127 A 0A        #0018:
0128 A 24        #0018:
0169 A           #0019:         RESERVE 64      /* reserve 32 stk levels */;
0169 A           #0020: STACK   EQU     $;
0169 A           #0021: SAVESP: RESERVE 2       /* saves sys stack pntr */;
016B A           #0021:
                 #0022:
016B A           #0023: ITER:   BYTE    0       /* counts iterations */;
016B A 00        #0023:
016C A           #0024: COUNT:  WORD    0       /* counts primes */;
016C A 0000      #0024:
016E A           #0025: PINDX:  RESERVE 2       /* points to pr. multiples */;
0170 A           #0025:
0170 A           #0026: PRIME:  RESERVE 2       /* prime number */;
0172 A           #0026:
0172 A           #0027: FLAGS:  RESERVE 8191    /* integers between */;
2171 A           #0027:
                 #0028:                         /*  1 & 8191 that are prime */


ChRoMoDassociates SMAL COMPILER, Vz4.3,          PAGE 0003
Sieve of Eratosthenes--SMAL/80

                 #0029:                         /*  0 => not prime */
                 #0030:                         /*  1 => prime */
1FFF A           #0031: FLAGLEN EQU     $-FLAGS         /* length of FLAGS */;
                 #0032:
                 #0033: START:
2171 A           #0034:         M(SAVESP)=SP    /* save system stack pntr */;
2171 A ED73      #0034:
2173 A 6901      #0034:
2175 A 316901    #0035:         SP=STACK        /* set up local stack */;
2178 A 110301    #0036:         DE=BGNMSG       /* DE -> message */;
217B A 0E09      #0037:         C=PSTRING       /* greet operator */;
217D A CD0500    #0038:         CALL BDOS;
                 #0039:
                 #0040:         LOOP            /* do 10 iterations */;
2180 A 210000    #0041:           HL=0          /* reset count */;
2183 A 226C01    #0042:           M(COUNT)=HL;
                 #0042:
2186 A 217201    #0043:           HL=FLAGS      /* initialize FLAGS array */;
2189 A 3601      #0044:           M(HL)=1       /* set first byte of move */;
                 #0044:
218B A 117301    #0045:           DE=FLAGS+1    /* DE -> destination */;
218E A 01FE1F    #0046:           BC=FLAGLEN-1  /* BC <- length of move */;
                 #0047:           LOOP;
```

➤

Figure 2 continued

```
                 #0048:             M(DE)=M(HL),
                 #0049:             ++DE,
                 #0050:             ++HL,
                 #0051:             --BC;
2191 A EDB0      #0052:           REPEAT UNTIL BC ZERO  /* BC now = 0 */;
                 #0053:                         /* use as FLAGS array index */
                 #0054:
                 #0055:           LOOP          /* check element in FLAGS */
2193 A DD21      #0056:             IX=FLAGS    /* IX -> base of FLAGS */;
2195 A 7201      #0056:
2197 A DD09      #0057:             IX=IX+BC    /* check element in FLAGS */;
2199 A DD7E      #0058:             A=M(IX+0)   /* A <- FLAGS element */;
219B A 00        #0058:
219C A 60        #0059:             H=B         /* HL <- FLAGS index */;
219D A 69        #0060:             L=C;
                 #0060:
219E A B7        #0061:             A=A OR A    /* prime? */;
                 #0062:             IF NOT ZERO THEN
219F A CADD21    #0063:               BEGIN     /*  yes */;
                 #0063:
21A2 A E5        #0064:                 PUSH HL         /* save index */;
21A3 A 29        #0065:                 HL=HL+HL        /* index to multiple */;
21A4 A 110300    #0066:                 DE=3            /* PRIME=indx+indx+3 */;
                 #0066:
21A7 A ED5A      #0067:                 HL=HL+DE+CY;
                 #0067:
21A9 A 227001    #0068:                 M(PRIME)=HL     /* save prime */;
21AC A D1        #0069:                 DE=POP          /* DE <- index */;
21AD A 19        #0070:                 HL=HL+DE        /* PINDX=index+prime */;
                 #0071: ZAPMULT:
21AE A           #0072:                 M(PINDX)=HL     /* multiples non-pr. */;
21AE A 226E01    #0072:
21B1 A B7        #0073:                 A=A OR A        /* clear carry */;
21B2 A 110020    #0074:                 DE=FLAGLEN+1    /* end of array? */;
ChRoMoDassociates SMAL COMPILER, Vz4.3,          PAGE 0004
Sieve of Eratosthenes--SMAL/80

21B5 A ED52      #0075:                 HL=HL-DE-CY;
                 #0075:
                 #0076:                 IF NOT CY
21B7 A 3018      #0077:                   (RELATIVE) GOTO INCCOUN   /*  yes */;
21B9 A DD21      #0078:                 IX=FLAGS        /*  no-- non-prime */;
21BB A 7201      #0078:
21BD A ED5B      #0079:                 DE=M(PINDX);
21BF A 6E01      #0079:
                 #0079:
21C1 A DD19      #0080:                 IX=IX+DE;
                 #0080:
21C3 A DD36      #0081:                 M(IX+0)=0;
21C5 A 00        #0081:
21C6 A 00        #0081:
21C7 A ED5B      #0082:                 DE=M(PRIME)  /* pnt to nxt multiple */;
21C9 A 7001      #0082:
```

➤

Figure 2 continued

```
21CB A 2A6E01    #0083:                 HL=M(PINDX);
21CE A 19        #0084:                 HL=HL+DE;
                 #0084:
21CF A 18DD      #0085:                 (RELATIVE) GOTO ZAPMULT;
                 #0086: INCCOUN:
21D1 A           #0087:                 IX=M(COUNT)      /* increment primes */;
21D1 A DD2A      #0087:
21D3 A 6C01      #0087:
21D5 A DD23      #0088:                 ++IX;
21D7 A DD22      #0089:                 M(COUNT)=IX;
21D9 A 6C01      #0089:
                 #0089:
21DB A 60        #0090:                 H=B              /* HL <- FLAGS index */;
21DC A 69        #0091:                 L=C;
                 #0092:               END;
21DD A 79        #0093:             A=C                  /* inc FLAGS index */;
21DE A C601      #0094:             A=A+1;
                 #0094:
21E0 A 4F        #0095:             C=A;
21E1 A 78        #0096:             A=B;
                 #0096:
21E2 A CE00      #0097:             A=A+0+CY;
21E4 A 47        #0098:             B=A;
21E5 A 11FF1F    #0099:             DE=FLAGLEN           /* end of array? */;
21E8 A B7        #0100:             A=A OR A             /* clear carry */;
                 #0100:
21E9 A ED52      #0101:             HL=HL-DE-CY;
                 #0101:
21EB A 20A6      #0102:            REPEAT UNTIL ZERO (RELATIVE);
                 #0102:
                 #0103:                                  /* process next */
                 #0104:                                  /* Iteration done. */
21ED A 3A6B01    #0105:            A=M(ITER)             /* A <- iterations */;
21F0 A 3C        #0106:            ++A                   /* inc iterations */;
21F1 A 326B01    #0107:            M(ITER)=A;
                 #0107:
21F4 A D60A      #0108:            A=A-10;
21F6 A C28021    #0109:         REPEAT UNTIL ZERO        /* do 10 times */;
                 #0110: /* Done--convert result to ASCII via brute force. */
21F9 A 2A6C01    #0111:          HL=M(COUNT)             /* show 4-digit */;
21FC A DD21      #0112:          IX=PRIMES               /*  ASCII decimal */;
ChRoMoDassociates SMAL COMPILER, Vz4.3,         PAGE 0005
Sieve of Eratosthenes--SMAL/80

21FE A 1A01      #0112:
2200 A DD36      #0113:          M(IX+0)='0'-1           /* init digit */;
2202 A 00        #0113:
2203 A 2F        #0113:
2204 A 01E803    #0114:          BC=1000                 /* convert thousands */;
                 #0115: CLOOP1:
2207 A           #0116:          ++M(IX+0)               /* simulate 1000 div */;
2207 A DD34      #0116:
2209 A 00        #0116:
```

Figure 2 continued

```
220A A B7          #0117:          A=A OR A               /* clear carry */;
220B A ED42        #0118:          HL=HL-BC-CY            /* less than 1000? */;
                   #0119:          IF NOT CY
220D A 30F8        #0120:            (RELATIVE) GOTO CLOOP1   /* no-keep going */;
220F A 09          #0121:          HL=HL+BC               /*  oops, too much */;
                   #0122:
2210 A DD23        #0123:          ++IX                   /* pnt to 2nd digit */;
2212 A DD36        #0124:          M(IX+0)='0'-1          /* initialize it */;
2214 A 00          #0124:
2215 A 2F          #0124:
2216 A 016400      #0125:          BC=100                 /* simulate 100 div */;
                   #0126: CLOOP2:
2219 A             #0127:          ++M(IX+0);
2219 A DD34        #0127:
221B A 00          #0127:
                   #0127:
221C A B7          #0128:          A=A OR A               /* clear carry */;
                   #0128:
221D A ED42        #0129:          HL=HL-BC-CY            /* less than 100? */;
                   #0130:          IF NOT CY
221F A 30F8        #0131:            (RELATIVE) GOTO CLOOP2   /* no-keep going */;
2221 A 09          #0132:          HL=HL+BC               /*  oops, too much */;
                   #0133:
2222 A DD23        #0134:          ++IX                   /* pnt to 3rd digit */;
2224 A DD36        #0135:          M(IX+0)='0'-1          /* initialize it */;
2226 A 00          #0135:
2227 A 2F          #0135:
2228 A 7D          #0136:          A=L                    /* do div by 10 */;
                   #0137: CLOOP3:
2229 A             #0138:          ++M(IX+0);
2229 A DD34        #0138:
222B A 00          #0138:
                   #0138:
222C A D60A        #0139:          A=A-10                 /* less than 10? */;
                   #0140:          IF NOT CY
222E A 30F9        #0141:            (RELATIVE) GOTO CLOOP3   /* no-keep going */;
2230 A C63A        #0142:          A=A+('0'+10)           /*  oops, too much */;
                   #0143:
2232 A DD23        #0144:          ++IX                   /* pnt to 4th digit */;
2234 A DD77        #0145:          M(IX+0)=A              /* save least digit */;
2236 A 00          #0145:
2237 A 111701      #0146:          DE=ENDMSG              /* DE -> message */;
223A A 0E09        #0147:          C=PSTRING              /* display results */;
223C A CD0500      #0148:          CALL BDOS;
223F A ED7B        #0149:          SP=M(SAVESP)           /* restore sys stk */;
2241 A 6901        #0149:
2243 A C9          #0150:          RETURN                 /* return to CP/M */;
2244 A             #0151:          END PROGRAM;
```

END FIGURE 2

lars balanced their cross-reference deficiency.

Of all the shortcomings, however, the lack of a linker and relocatable code will probably be cited by many software jocks as the *coup de grace.* This limitation might be overlooked in a low priced bargain assembler. But to demand $250, the second highest price of the eight assemblers, and not include a linker . . . I think David Stockman would frown on that.

Despite the slow speed, if price is a primary factor in your choice of an assembler or if you resort to assembly language on rare occasions for relatively small files, then the 2500AD package may be the best deal in town.

Well Documented

What software is included with the package, is well described in the user's manual. On the credit side of the ledger, the documentation ranks next to performance. The seven-chapter manual covers operational and functional aspects from head to toe. Included is a chapter summary of the Z80 instruction set. The text is well organized and reads clearly, but quick referencing is hampered because the authors left out an index.

Overall, Avocet's assembler has several outstanding qualities. But missing components detract significantly from its suitability as a heavy duty software tool. At a lower price, the shortcomings could be overlooked in the interests of economy, but at $250 . . . eschew.

2500AD: Slow But Robust

Three months ago I would have recommended 2500AD's Z80 macro assembler without reservation. Priced then at $49.50, it sported a list of features that compared favorably with the thoroughbreds of the assembler world. It did come in far behind the pack during execution, but over 1100 users balanced that with the inexpensive cost.

Though 2500AD's assembler still posts a poor showing when gauged with a stopwatch, the price has now been doubled. At $99.50, the product approaches the cost bracket of high powered assemblers. In this league, prospective buyers may opt to spend an extra C-note to purchase faster software.

Just how wide is the gap between slow and fast? The statistics speak for themselves. The SLR Systems program holds the title of fastest in this roundup of eight assemblers. It processes the 148 line Sieve benchmark in 23 seconds. The 2500AD assembler takes 127 seconds to perform the same task, a factor of over five times greater.

For short programs, the difference may not amount to a significant wait. If you consider a sizeable program, however, the performance margin can grow intolerable. For example, what takes the SLR assembler 10 minutes to complete will require the 2500AD program about an hour. Simple arithmetic shows that the larger investment in speedy software will be recovered in short time with less wages paid to programmers.

Mandatory Link

Part of the slow performance is attributable to the linker. Creation of an executable (".COM") file requires invocation of the linker even if the source statements generate absolute, nonrelocatable code. Regrettably, the 2500AD assembler resembles Microsoft's in this respect. In the Sieve benchmark, half of the 127 seconds occurs during the link step.

Users have the option of creating ".COM" (CP/M executable), ".HEX" (Intel hexadecimal) or ".TEK" (Tektronix hexadecimal) files from the link. This is accomplished by invoking program LINK1, LINK2 or LINK5. Each of these tallies to approximately 15K on the single diskette on which the software is delivered.

Despite the slow speed, there is a market for budget priced software. If price is a primary factor in your choice of an assembler or if you resort to assembly language on rare occasions for relatively small files, then the 2500AD package may be the best deal in town.

Referring to the comparison chart, you can see that this assembler holds its own feature for feature against the competition. These features include relocatable code, interactive mode and cross-reference generator. The cross-reference generator is wisely built into the 30K assembler and does not entail an external program as do the Digital Research and Microsoft products.

The 60 page user's manual lists a generous selection of assembler directives. The more notable ones are ASK, RADIX and MOD32. ASK prompts the console for a one character response during assembly. This response may be used for subsequent conditional assembly.

RADIX allows assembly-time arithmetic to be set to the usual binary, octal, decimal or hexadecimal number bases.

32-Bit Precision

With MOD32, 2500AD has an arm up on the competition. This directive specifies 32-bit precision for assembly-time arithmetic in lieu of the usual 16-bit. Coupled with a robust repertoire of operators including exponentiation, the assembler takes the drudgery out of calculating arithmetic expressions. (Incidentally, the benchmarks were performed in 16-bit mode, the default.)

Other assembler directives include full listing controls: page title and sub-title, page length and width, and selective printing of the source code.

During execution, the assembler responds to over a half-dozen keyboard commands. These cover typical imperatives such as route listing output directly to the printer, abort the assembly, and temporarily halt the assembly until instructed to continue. One which I found particularly useful was the command sequence to dynamically display the assembly on the terminal screen. Control-`T` turns on the display and Control-`N` turns it off. This allows you to spot check the progress of an assembly without incurring the overhead of scrolling the entire file on the terminal.

Extras

In addition to the assembler and linker, 2500AD bundles three other programs with the package. The first, CONV, converts Intel 8080 source code mnemonics to Zilog Z80 mnemonics. This handy tool works as advertised with one exception. It mysteriously converts a `POP DE` instruction into `POP DEE`, the only vestige of a bug in the entire system.

Program CASE translates lower case characters in a source code file to upper case. Strings enclosed by apostrophes are ignored. I'm certain no one is going to buy the package for this program alone, but it's there for the discerning user.

Finally, a program called HEX rounds out the directory of extras. HEX performs the inverse of CP/M's LOAD utility. It creates an Intel hexadecimal format ".HEX" file from an executable ".COM" file.

To summarize, 2500AD Software Incorporated delivers a robust package saddled with dismal performance. If your computing budget leaves only $100 for an assembler, by all means pursue 2500AD's package. But if you don't want to keep your CP/M system up late at night churning through assemblies, then spring for a higher priced brand. After all, it was Ben Franklin who once observed, "Time is money."

Z sets you FREE!

Z — yes! Synergistic combination of ZCPR3 and ZRDOS2 produces flexible state-of-the-art Z80 operating system with tremendous productivity features.

Z-System consists of software modules, dynamic loading segments, and tools permitting optimum computer usage ranging from production program development to turnkey, password-controlled, end-user installations. Facilities include: multiple commands per line, file search paths, named directories, I/O redirection, command flow control, screen-oriented menu generators, complete housekeeping file and directory management, shells, alias (scripts) and nested-alias generation, and complete online help.

Seventy-six support utilities, five tool packages, and two application programs available now! Fully upward compatible with CP/M-80.

Z can now be purchased as auto-install program (Z-Com) or as manual-install ZCPR3 with semi-auto install ZRDOS package (Z-System). Our latest versions, to be released this year, support Zilog Z800 and Hitachi HD62801/64180 high-technology chips, chips run existing 8080 and Z80 programs!

Echelon eight-bit operating systems written in Assembly Language, using linkable macro subroutine libraries, offer performance paralleling best single-user 16/32-bit microcomputer systems.

1. **Z-Com** Full-up Z Operating System with input/output redirection running under CP/M-80, online command and utility documentation and help system **$219.95**

2. **Z-System** Manual-install ZCPR3 and ZRDOS2, easily tailored by programmer to custom needs; source code to core and utilities; similar to Item 1 .... **$199.95**

3. **Z-Tools** Four software development system packages permitting advanced, structured program design, macro relocating assembler, linking loader, librarian, cross-reference generator, debugger, mnemonic and pseudo-op translators, and interactive disassembler. Super $315.00 package value **$200.00**

4. **DSD** Dynamic Screen Debugger offers high-level features never before found in microcomputers; simultaneous display of dual-memory segments, stack, cpu states, and flags, with software In-Circuit-Emulation **$149.00**

5. **The Libraries** Linkable ZCPR3 libraries (Vlib, Z3lib, and Syslib3) of over 400 subroutines used for Assembly Language program writing. Simplifies structured, efficient code production; online help system and full source code provided **$45.00**
Syslib3 alone **$29.00**

6. **Term3** New generation communication program permits menu control of computer/modem operations between operator and time-share services, bulletin-boards and other remote computer systems; auto-answer to command-line prompt **$99.00**

7. **Discat** Fancy file and disk catalog program running under Z-System, menu driven and easily customized by operator **$49.00**

Fortnighter newsletter, 24-hour BBS Z-Node System keep Z users informed of microcomputer happenings. Write or call for brochure or order now! State disk format desired; add $3.00 shipping & handling; Californians please add 6-1/2% sales tax. Visa/MC, check, money or purchase order accepted. (Program names are trademarks of their respective owners.)

Echelon, Inc.

101 First Street • Los Altos, California 94022 • 415/948-3820

Circle number 4 on the reader service card.

New Products

CP/M and MS-DOS computers, software, communications products, networks and accessories

Free information is available on the products described here. We provide a *reader service card* in this issue with numbers corresponding to each product announcement. Simply fill out the reader service card and drop it in the mail (please include your address). We forward your address to the product manufacturer or vendor, who sends you the information.

Thanks for filling out the questionaire on the card. This feedback helps us plot an editorial course, and the answers provide demographic information to media planners. The more responses this magazine can generate, the more it can grow in pages and in product coverage. We appreciate your support.

The press announcements in this section have not been checked for accuracy, and we cannot verify that the products are available at this time. In most cases we do not have the products in-house, and they have not yet been reviewed in this magazine. Therefore, take claims of product features and benefits with a "grain of salt."

Computers and Systems

The Discovery

The Discovery 1600 multiprocessor (Action Computer Enterprise, Inc.) is a multi-user microcomputer system that allows up to 16 users to share common resources. Both CP/M Plus and CP/M-86 (from Digital Research Inc.) are supported by dpc/os, allowing the simultaneous operation of both 8-bit and 16-bit users in the same system.

Completely compatible with existing CP/M and Concurrent CP/M (PC-DOS emulation) software, version 3.0 of dpc/os also allows the networking of IBM PC and compatible workstations. Discovery users can run without modification thousands of 8-bit and 16-bit applications programs written for CP/M and the IBM PC.

Multi-user software can be developed easily on this IEEE 696/S-100 bus system, which has 20 slots in the Rack-Mount (36-inch high cabinet with walnut grain top) configuration. The standard Table-Top configuration makes a good starter system.

Each user is provided with an 8-bit (CP/M Plus) or 16-bit (CP/M-86 and Concurrent DOS) environment. The system supports any combination of these environments with shared peripherals and data, file and record locking and interlocking processor communications.

For price and delivery schedules, contact Action Computer Enterprise Incorporated, 430 N. Halstead St., Pasadena, CA 91107. (818) 351-5451, (800) 821-6596 outside CA.

Circle 51 — reader service card.

Epson's New "Notebook" Computer

The Z80-based Geneva/PX-8 with 64K RAM and 32K ROM includes CP/M, an "expanded" BASIC language and Micropro's word processing, spreadsheet and scheduling software on ROM. Portable WordStar is compatible with standard CP/M WordStar, and has most of the desirable features (copy block movement, continuous word wrap, margin justification) that WordStar users expect. Portable Scheduler provides appointment scheduling and Portable Calc provides spreadsheet functions.

Peripheral options developed for the PX-8 allow expansion to 184K RAM in 60K and 120K expansion units. A "multi-unit" combines a direct-connect modem (no baud rate specs. available at press-time) with a 64K expansion RAM disk, expected to ship mid-August. The $3^1/_2$" battery-operated floppy disk drive (with 320K storage capacity per disk) has no specified ship date, but may ship with the PX-8 at the end of July. A 9-pin 80-column thermal transfer printer will ship mid-September.

Any Epson dot matrix printer can be serially interfaced to the PX-8. The four-pound 8$^1/_2$" by 11$^1/_2$" by 1$^3/_4$" PX-8 sells for a base price of $995. Included in the base unit is an 8-line-by-80-character high-resolution LCD display with adjustable viewing angle, a full-sized ASCII keyboard with numeric keypad functions, a built-in microcassette tape drive for data storage, resident CP/M and BASIC, and ROM based Micropro applications software. The PX-8 is powered by rechargeable Ni-cad batteries that provide up to 20 hours of operating time between charges. Epson America, Torrance, CA.

Circle 52 — reader service card.

Software

Professional Help From Hugh

Data Based Solutions announced the "Hugh T. Birckhead Professional Line of Database Software," consisting of four $35 programs. These inexpensive programs complement dBASE II and are written in C.

dVALIDATE generates "bullet proof" data entry validation for every field in your dBASE II data bases. dEFLATE cures dBASE II command file bloat, saving 40% in disk storage space.

dLINEATE offsets DO WHILE, IF and DO CASE pairs, then draws a vertical line between the beginning and end of each set; it also prefixes each format error with a # and moves it to the left margin. dQ is a quick search/find/report utility which makes it possible for an inexperienced dBASE II user to easily manipulate a database.

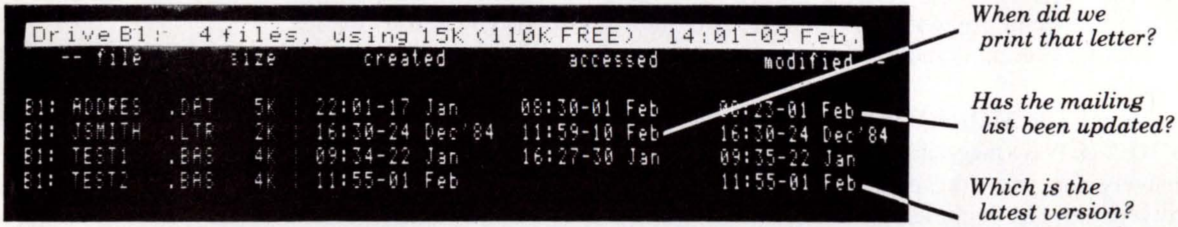
Data Based Solutions, 1975 Fifth Avenue Suite 105, San Diego, CA 92101. (619) 236-1182.

Circle 53 — reader service card.

English/Hebrew Word Processor

Wordmill is a word processor that offers screen menus, commands and documentation in English or Hebrew.

DATESTAMPER™ has the answers



```
Drive B1:  4 files, using 15K (110K FREE)  14:01-09 Feb.
  -- file        size        created          accessed         modified --

B1: ADDRES  .DAT   5K  22:01-17 Jan     08:30-01 Feb     08:23-01 Feb
B1: JSMITH  .LTR   2K  16:30-24 Dec'84  11:59-10 Feb     16:30-24 Dec'84
B1: TEST1   .BAS   4K  09:34-22 Jan     16:27-30 Jan     09:35-22 Jan
B1: TEST2   .BAS   4K  11:55-01 Feb                      11:55-01 Feb
```

When did we print that letter?
Has the mailing list been updated?
Which is the latest version?

DateStamper™ keeps your CP/M computer up-to-date!

- avoid erasing the wrong file
- back-up files by date and time
- keep dated tax records of computer use
- simplify disk housekeeping chores

OPERATION: DateStamper extends CP/M 2.2 to automatically record the date and time a file is created, read or modified. DateStamper reads the exact time from the real-time clock, if you have one; otherwise, it records the order in which you use files. Disks initialized for datestamping are fully compatible with standard CP/M.
INSTALLATION: Default (relative-clock) mode is automatic. Configurable for any real-time clock, with pre-assembled code supplied for popular models. Loads automatically at power-on. **UTILITIES:** • Enhanced SuperDirectory • Powerful, all-function DATSWEEP file-management program with date and time tagging • Disk-initializer • Installation and configuration utilities **PERFORMANCE:** Automatic. Efficient. Invisible. Compatible.

Requires CP/M 2.2. Uses less than 1K memory. Real-time clock is optional.

When ordering please specify format

8" SSSD, Kaypro, Osborne Formats	**$49**
For other formats (sorry, no 96 TPI) add $5.	
Shipping and handling	**$3**
California residents add **6%** sales tax	

MasterCard and Visa accepted

Specialized versions of this and other software available for the Kaypro.
CP/M is a registered trademark of Digital Research, Inc.

Write or call for further information

Plu*Perfect Systems

BOX 1494 • IDYLLWILD, CA 92349 • 714-659-4432

Circle number 15 on the reader service card.

Text can be entered via the keyboard, with left-to-right or right-to-left cursor movements entered automatically, in English or Hebrew. You switch between languages with a single keystroke, so it is possible to edit in both languages simultaneously.

Wordmill features generation of 90 special scientific characters, Greek letters and mathematical symbols for scientific/mathematical text processing, as well as the usual file operation, editing, printing, and data exchange capabilities.

Wordmill is available for the IBM PC, XT, AT and Portable, Compaq and DEC Rainbow. Other Wordmill versions use English, French, German, Greek, Hebrew and other languages.

The Omnigal bilingual word processor package for the IBM PC and Epson FX-80 printer contains two IC chips, which you install in your computer and printer so they can display and print the Hebrew characters, a keyboard template kit, the Wordmill program on disk and documentation.

For price and other configurations, contact Omnigal Systems, 4170 Wallis Court, Palo Alto, CA 94306. (415) 394-0463 or 856-8177.

Circle 54 — reader service card.

Turbo Pascal 3.0

Borland International has released a new version (3.0) of Turbo Pascal for CP/M-80, CP/M-86, MS-DOS and PC-DOS.

One keystroke will run and compile a program and provide fast results while eliminating the need to call a separate linker program, thanks to the new one-step compile feature that integrates linker and compiler. Version 3.0 is more than twice as fast as 2.0 in compile and execution speeds, and requires approximately 35K of memory.

The IBM versions (PC-DOS and MS-DOS) also feature a memory-mapped editor (full-screen, with WordStar-like commands) for more efficient file, block and page moves. Turtle graphic functions, added to the IBM versions, are particularly useful in pie and bar chart graphics.

The file input/output system now performs I/O redirection, and handles some of the functions implemented in often used DOS filters (MORE and SORT, for example).

MS/PC-DOS tree structured directory support and path name support are provided, and overlay files can be used on any path or any drive.

Version 3.0 can be ordered for $69.95 until June 1, 1985. For Turbo Pascal 3.0 with either 8087 or BCD support options, add $39.95; with both add $55.

Current Borland customers can trade in their Turbo 2.0 original disk for a $39.95 credit toward purchase of 3.0, or their Turbo 2.0 with 8087 support original disk for a $59.95 credit towards purchase of the new Turbo

If you own a Morrow . . . then you need MOR

Tired of searching newsstands and bookstores for *something* that relates to your Morrow?

Here it is! The **MORROW OWNERS' REVIEW** is a magazine published exclusively for you—the owner of a Morrow computer. Beginners as well as technical wizards can find high-quality how-to articles, reviews of books and software, tips on new products, tutorials—information you won't find elsewhere, written by Morrow owners for Morrow owners. Our columnists will answer those questions you're too embarrassed to ask at your local users group (if you have one). And our display advertisers tell you only about products for your Morrow computer.

$18 a year is all it takes to get **MOR** delivered to your front door every other month. That's less than the cost of a box of cheap floppies!

Back issues are available for $3.00 each, including postage. (Five issues of Vol. 1 published in 1984; bimonthly in 1985, starting with Vol. 2 in February.)

Morrow Owners Review

Name ______________________

Address ______________________

City ____________ State ______ Zip ____________

☐ One year $18.00 ☐ Two years $32.00

☐ Check enclosed ☐ MasterCard ☐ VISA

Card # ______________________

Expiration Date ______________________

Signature ______________________

Send to: **Morrow Owners' Review**
Box 5487, Berkeley, CA 94705, *Attn: Subscriptions*

SPECIAL MAILING COSTS: For First Class, add $10/yr. for US, Canada & Mexico. For all other countries, add $10/yr., for surface mail, add $40/yr. for Airmail.
TUG685

Circle number 8 on the reader service card.

with 8087 or BCD options. The trade-in offer is available directly through Borland, and expires June 1, 1985.

Turbo Pascal 3.0 is available to first time purchasers through Borland or through dealers nationwide. Call (800) 556-2283 for a dealer in your area. Credit card orders to Borland, call (800) 225-8008, or in California (800) 742-1133.

Borland International, 4113 Scotts Valley Drive, Scotts Valley, CA 95066. (408) 438-8400.

Circle 55 — reader service card.

Terminals

KDT 480 Mobile Data Terminal

Motorola Inc., Information Services announces the KDT 480 mobile data terminal with storage of up to 3000 characters in dynamically allocated RAM.

The CRT display accommodates up to 480 resolution, with 12 lines of 40 characters each displayed, plus two 80-character status lines, for a total text area of over 20 square inches. The terminal display was designed for varying light conditions in a moving vehicle, so visibility is assured, even in direct sunlight.

The full-sized typewriter-style keyboard has color-coded function keys. An emergency indicator enables a driver to transmit an alert by pressing a function key or remotely mounted switch.

The KDT 480 terminal can be used as a dedicated radio system or it can be incorporated into an existing radio system. Field personnel can access computer networks from remote locations, customer sites and even after routine business hours for maximized efficiency in a mobile environment.

Pricing, features, options and additional information is available from Motorola, Inc., Communications Sector, Public Relations Dept., 1301 E. Algonquin Road, Schaumburg, IL 60196.

Circle 56 — reader service card.

Displayphone PLUS

Northern Telecom Inc. first announced the Displayphone (terminal and phone) in 1981, with a 300 baud

New Products *continued on page 72*

Checks & Balances™

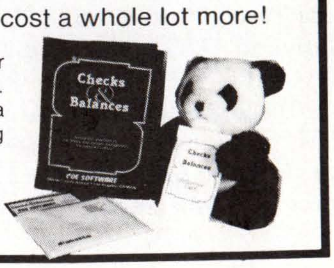
is creating Panda-monium for Home Accountant and Quick Check!

When people compare Checks & Balances to **Home Accountant** and **Quick Check,** it's no contest! Just look at these examples:

- **We** give you powerful, easy to use and easy to remember English commands to take you from one function to another instantly. **They** give you a menu to go to another menu which goes to another menu to go to another menu. . . .
- **We** put up to seven entries on a screen at a time in a form just like your check register and let you scroll forward and back like flipping pages. **They** disply one entry per screen that doesn't look at all like your familiar check register.
- **We** let you make corrections on any entry displayed by inserting or deleting characters or typing over the error just like your word processor. With **them** making corrections means going to special menus where you have to call up the entry again, then going back through the menus to resume your work.
- **We** give you over forty characters of space for the payee and memo instead of as few as fifteen.
- **And best of all . . . we** cost only **$74.95. They** cost a whole lot more!

CHECKS & BALANCES also has a Rolodex for names, addresses, phone numbers and notes, can print nearly any type of check, has reports comparing income and expenses, lets you disperse any entry to multiple categories, keeps a full year at a time, and gives you as many checkbooks as you want, on any disk drive.

Start the New Year out right (and get last year in order for tax time). Pick up **CHECKS & BALANCES** at your local dealer, or order direct from CDE. **CHECKS & BALANCES** is available for PC-DOS, MS-DOS and CP/M. Indicate computer type, operating system, and disk type (8" SSSD, 5¼"SSDD, or 5¼" DSDD). California residents include 6% sales tax, and everyone include $2 shipping and handling. Dealer distribution bundling rates available. Write for complete catalogue of software for home and business.



Visa/MC accepted **CDE SOFTWARE** (213)-661-2031

2463 McCready Ave. • Los Angeles, CA 90039

Circle number 2 on the reader service card.

CP/M-80 •
CP/M-86 •
PC-DOS •
MS-DOS •

Turbo Pascal, Part 2: Text Conversion

This installment describes programming techniques and introduces a Pascal program that converts text printing codes into typesetting codes.

by Tony Bove & Cheryl Rhodes

Turbo Pascal is available from Borland International, 4585 Scotts Valley Drive, Scotts Valley, CA 95066. Version 3.0 is $69.95 ($39.95 trade-in credit given to holders of earlier versions).

We had intended to cover mathematical functions and structured programming in this second installment, but Turbo Pascal version 2.0 has minor "bugs" (software errors) relating to floating point arithmetic. We are scheduled to receive version 3.0, which is supposed to have (among other things) bug fixes, so we've put off our discussion of math functions until then.

For a list of known bugs in Turbo Pascal, check out the newsletter from the Turbo Users Group (TUG, no relation to TUG Inc.) called *TUG Lines* (P.O. Box 1510, Poulsbo, WA 98370).

Turbo Pascal is the first non-BASIC programming language we can recommend for beginners. The Pascal programming language has a recognized standard implementation that is taught in schools, and Turbo Pascal is close enough to the standard to make it easy to learn.

We see Turbo Pascal as an all-purpose problem solving tool. For amateur programmers (those who don't make a living at it), Turbo Pascal has the power of a professional programming tool at an affordable price ($69.95).

Part one of this series (issue #10) described the Turbo Pascal compiler and how to edit a program. The editing commands are very similar to WordStar's editing commands — some are exact replicas, others are similar enough that a WordStar user could try Turbo Pascal's editor without reading the documentation.

The first part included an example of a real problem to solve: converting a data base file from a strange record format to a familiar one compatible with MailMerge and dBASE II. That installment introduced various Pascal statements (READLN, WRITELN, WHILE/DO, BEGIN/END, ASSIGN), string assignments, and the CONCAT function. It also gave step-by-step instructions on how to edit, compile and run a Turbo Pascal program. You should go back to that installment if you have never done this.

This installment assumes you know how to edit, compile and run a Turbo Pascal program. It also assumes you know how to write a simple program in Pascal using BEGIN/END, READLN, WRITELN, WHILE/DO, and ASSIGN statements, string assignments, and the CONCAT function. We introduce a little *programming style* in this installment (including use of "procedures"), and provide an example of customizing your text processing.

The Problem

We have another real problem to solve with amateur programming skills. The first step in any problem-solving venture is to define the problem. It helps to put it all in one sentence — this forces you to see the highest logical connection.

Here's the problem: we need a program to quickly convert our WordStar-coded text into typesetting-coded text suitable for computerized phototypesetting.

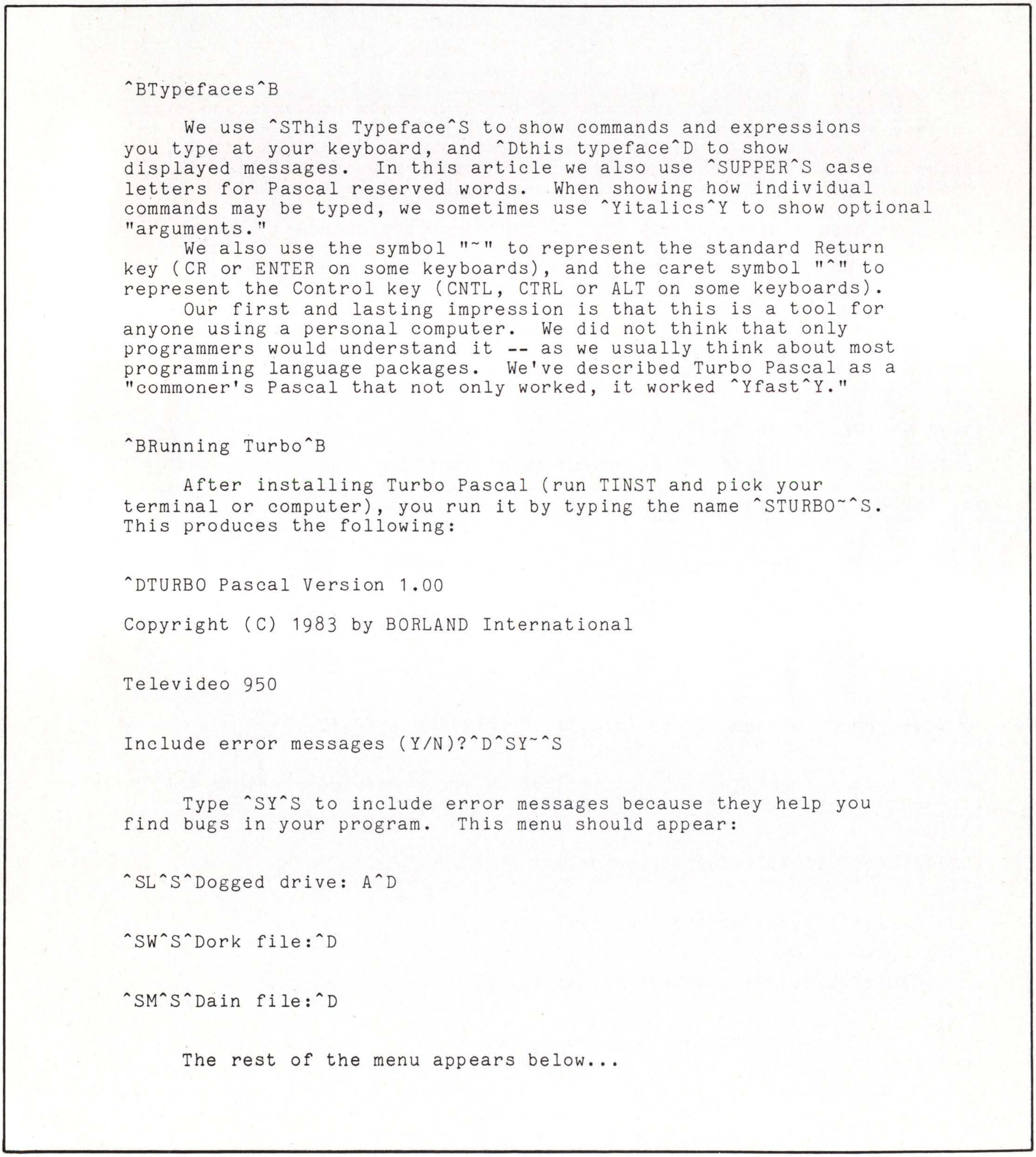
Here are some details, starting with *what the output should look like.*

The typesetting computer uses a typesetting language consisting of codes embedded in the text. These codes control typeface changes, indenting, margins, justification — virtually *any* typesetting feature can be controlled directly with these codes.

Why not type the codes directly into the text we type? They are clumsy-looking (which slows down proofreading) and prone to error (easily typed incorrectly, adding to the proofreading task). Typical change typeface codes looks like this:

```
{cf2,10,12}
```

Figure 1. *Sample text with WordStar printing codes, indented paragraphs, ordinary quotation marks, and blank lines, ready to be processed.*



```
^BTypefaces^B

     We use ^SThis Typeface^S to show commands and expressions
you type at your keyboard, and ^Dthis typeface^D to show
displayed messages.  In this article we also use ^SUPPER^S case
letters for Pascal reserved words.  When showing how individual
commands may be typed, we sometimes use ^Yitalics^Y to show optional
"arguments."
     We also use the symbol "~" to represent the standard Return
key (CR or ENTER on some keyboards), and the caret symbol "^" to
represent the Control key (CNTL, CTRL or ALT on some keyboards).
     Our first and lasting impression is that this is a tool for
anyone using a personal computer.  We did not think that only
programmers would understand it -- as we usually think about most
programming language packages.  We've described Turbo Pascal as a
"commoner's Pascal that not only worked, it worked ^Yfast^Y."


^BRunning Turbo^B

     After installing Turbo Pascal (run TINST and pick your
terminal or computer), you run it by typing the name ^STURBO~^S.
This produces the following:


^DTURBO Pascal Version 1.00

Copyright (C) 1983 by BORLAND International


Televideo 950


Include error messages (Y/N)?^D^SY~^S


     Type ^SY^S to include error messages because they help you
find bugs in your program.  This menu should appear:


^SL^S^Dogged drive: A^D


^SW^S^Dork file:^D


^SM^S^Dain file:^D


     The rest of the menu appears below...
```

This changes the typefont to font 2, at 10 point on 12 leading. The codes make text difficult to read; besides, they might have to change if we wanted to take the file to a different typesetting service. For this reason alone you should consider solving this problem if you intend to have a lot of text typeset or printed with different typefaces on a laser printer.

We wanted to use our WordStar print control codes (as defined for our Epson MX-100 printer) to control typeface changes, and then have a quick program change these codes into the typesetting codes. The advantage would be that the manuscripts could

Figure 2. *The processed file after running the program shown in figure 4; it now has typesetting codes replacing the printing codes.*

```
{cf4}Typefaces{cf2,10,12}
{ql
{mc}   We use {cf396}This Typeface{cf2,10,12} to show commands and expressions
you type at your keyboard, and {cf397}this typeface{cf2,10,12} to show
displayed messages.  In this article we also use {cf396}UPPER{cf2,10,12} case
letters for Pascal reserved words.  When showing how individual
commands may be typed, we sometimes use {cf3}italics{cf2,10,12} to show optional
"arguments.''
{mc}   We also use the symbol "~'' to represent the standard Return
key (CR or ENTER on some keyboards), and the caret symbol "^'' to
represent the Control key (CNTL, CTRL or ALT on some keyboards).
{mc}   Our first and lasting impression is that this is a tool for
anyone using a personal computer.  We did not think that only
programmers would understand it -- as we usually think about most
programming language packages.  We've described Turbo Pascal as a
"commoner's Pascal that not only worked, it worked {cf3}fast{cf2,10,12}.''
{ql}
{ql}
{cf4}Running Turbo{cf2,10,12}
{ql}
{mc}   After installing Turbo Pascal (run TINST and pick your
terminal or computer), you run it by typing the name {cf396}TURBO~{cf2,10,12}.
This produces the following:
{ql}
{ql}
{cf397}TURBO Pascal Version 1.00
{ql}
Copyright (C) 1983 by BORLAND International
{ql}
{ql}
Televideo 950
{ql}
{ql}
Include error messages (Y/N)?{cf2,10,12}{cf396}Y~{cf2,10,12}
{ql}
{ql}
{mc}   Type {cf396}Y{cf2,10,12} to include error messages because they help you
find bugs in your program.  This menu should appear:
{ql}
{ql}
{cf396}L{cf2,10,12}{cf397}ogged drive: A{cf2,10,12}
{ql}
{ql}
{cf396}W{cf2,10,12}{cf397}ork file:{cf2,10,12}
{ql}
{ql}
{cf396}M{cf2,10,12}{cf397}ain file:{cf2,10,12}
{ql}
{ql}
{mc}   The rest of the menu appears below...
{ql}
```

be printed on the Epson for proofreading, and then be converted for typesetting.

In addition to the typeface change codes, we needed to add an "end of paragraph" *merge code* for controlling paragraph ends and indented first lines, and we needed to add a "quad left" code to force single lines or ends of paragraphs to be justified to the left. We also needed to change *every closing quote* (' ' ' ') of a quoted word or phrase to two apostrophes ("), because the typesetting unit uses two different characters (") and (") to enclose quotations. All these problems are handled by our Turbo Pascal program.

We use the following WordStar printer control codes:

`^Y` — Turn on/off italic (font 3).
`^B` — Turn on/off bold (font 3).
`^S` — Turn on/off Special #1 "example" font (font 397).
`^D` — Turn on/off Special #2 "display" font (font 396).

These codes disappear from your editing screen when you type the `^OD` command, although they remain in your file. They can be proofread by printing the text or by typing `^OD` again, which brings them back into view. The only restriction is that you must use the same code to *turn off* the typeface as you used to turn it on.

Typesetting codes do not work this way: to *turn off* one typeface you must select another. The "turn off" code for all the above typefaces is `{cf2,10,12}`, which we must substitute for the "turn off" codes above. Somehow the program must be able to tell if a `^B` is turning *on* bold or turning it *off*.

To handle the merge code problem, we consistently indent the start of each paragraph with five spaces; when the program sees a line starting with five spaces, it automatically substitutes a merge code.

To put "quad left" (`{ql}`) codes at the end of single lines and the end of a block of paragraphs, we have the program insert a "quad left" whenever it finds a blank line (a line with nothing but a single Return). Therefore, we can consistently type two Returns after every single line to have a "quad left" appear there.

These measures may seem arbitrary and unnecessary to someone who doesn't prepare text for typesetting on a regular basis. However, they shorten our processing time by several hours. We would otherwise have to use WordStar's search and replace function (or similar program) and answer `Y` or `N` to a lot of substitutions. With this program on hand, we can type ordinary WordStar text, with indented paragraphs, two Returns after single lines, and printer control codes, and in one easy step convert the file into a format for typesetting.

We attribute this good fortune to Turbo Pascal, because we would never have attempted such a program in BASIC (except perhaps a structured BASIC like CB-80). This type of program is possible to write in MBASIC or CBASIC (two very popular languages bundled with computers), and you may be able to write one in assembly language that is smaller and runs faster, but for the combination of simplicity, performance, and inexpensive computing, you can't beat Turbo Pascal. You don't need to be an expert in programming to work with this language.

The Solution

Figure 1 shows sample text with WordStar printing codes, indented paragraphs, ordinary quotation marks, and blank lines, ready to be prepared for typesetting.

Figure 2 shows the result of running the program (listed in figure 4), and figure 3 shows the typeset result.

The program in figure 4 is complete and should run without modification if you type it in as shown. Comments are shown within {* and *} brackets; the rest of the text in figure 4 are Pascal statements. We use all `UPPER` case for Pascal words and the combination of Upper/lower case for our made-up variable and procedure names.

Figure 3. *The result of typesetting the file in figure 2.*

Typefaces

We use `This Typeface` to show commands and expressions you type at your keyboard, and **`this typeface`** to show displayed messages. In this article we also use `UPPER` case letters for Pascal reserved words. When showing how individual commands may be typed, we sometimes use *italics* to show optional "arguments."

We also use the symbol "↵" to represent the standard Return key (CR or ENTER on some keyboards), and the caret symbol "^" to represent the Control key (CNTL, CTRL or ALT on some keyboards).

Our first and lasting impression is that this is a tool for anyone using a personal computer. We did not think that only programmers would understand it — as we usually think about most programming language packages. We've described Turbo Pascal as a "commoner's Pascal that not only worked, it worked *fast*."

Running Turbo

After installing Turbo Pascal (run TINST and pick your terminal or computer), you run it by typing the name `TURBO`↵. This produces the following:

```
TURBO Pascal Version 1.00
Copyright (C) 1983 by BORLAND Intl.

Televideo 950

Include error messages (Y/N)?Y↵
```

Type `Y` to include error messages because they help you find bugs in your program. This menu should appear:

```
Logged drive: A

Work file:

Main file:
```

The rest of the menu appears below...

The Algorithm

A Pascal program should start to form in your mind as a complete solution, from the "top down" (to use data processing lingo). We started with a one-sentence description; now we'll break it up into tasks:

1. Open the text file and create a new output file; set up the process to occur from the beginning to the end of the text file.

2. Read each line; if the line is not blank, then check to see if it starts with a space; if so, check the next three characters to see if they're spaces, and if so, substitute a merge code for the spaces; either way, skip the space or spaces.

3. From the beginning to the end of the input line, check each character: if it is in the set of printing control characters, substitute the appropriate typesetting code; if it is a double-quote, substitute the appropriate open or close quote symbols. Determine the appropriate code or symbol by keeping track of what the *last* one was (starting with a "turn on" code and an open quote).

4. If the character is not one of the above, add it to the output line to be sent to the output file, and advance to the next character in the input line.

5. If the line is a blank line (does not satisfy #2 above), insert a quad left code into the output line (ignoring the blank input line), and move on.

6. Display each output line and send it to the output file, and clear the output line to receive the next input line; when finished, close both files.

It may take a while to think in such rigid logical steps, but this process makes it much easier to write a Pascal program.

The next step is to examine the major tasks: opening and creating files, checking for spaces and substituting merge codes, and substituting typesetting codes for WordStar print codes. These major tasks are single statements in our algorithm because we haven't yet defined them.

Smaller tasks that frame the entire problem are checking for blank lines, and checking to see if the character is in the set of print codes. These tasks are spelled out in the algorithm in IF/THEN statements that control the structure of the program.

The final, missing, element is the method by which the program can determine whether the print code is turning *on* a typeface or turning it *off*, and whether the double-quote character is representing an "open quote" or a "close quote".

The program must "learn" from its own operation whether or not a print code is an "on" or an "off" code.

We devised a simple method to handle both problems. The program must "learn" from its own operation whether or not a print code is an "on" or an "off" code. When the program first encounters a print code, it assumes the code is turning the typeface *on.* When it next encounters a print code, it assumes the code is turning the typeface *off.*

Since we know that WordStar print codes are the same for "on" and "off" (they are appropriately called *toggles*), we can get by with one "assumption condition" that tells the program whether to use "on" or "off" for typesetting codes. This condition is set to true or false by using a *boolean variable* for the condition: `NextOne`.

We use another boolean variable, `NextQuot`, to establish the same type of condition for open and close quotes. In this case we must set up a rule for our text files: we cannot use the double-quote symbol for anything other than opening and closing quotes, and they must be used in pairs. If we need to use the closing quote symbol by itself (for example, to represent two inches as 2"), we must instead use two apostrophes. This is a rule we can live with, since it was a convention we used anyway.

Constants and Variables

One of the pleasures of using Pascal is the ease with which you can define the constants as well as the variables you are using in your program. You define them once, and the program uses the defined words to perform operations. If you want to change the constants, you simply change the definitions, and the program works fine with the new definitions.

Rather than burying our printing and typesetting codes in the body of the program, we define them as constants at the top of the program where they can be easily changed for different typesetters or printers.

The printer codes are expressed using the caret symbol (^) and the letter of the code. Control-`PY`, for example, produces a Control-`Y` character in the text that turns on or off the Epson printer's italic typeface. To express Control-`Y` as a constant in a Turbo Pascal program, you type a caret followed by `Y` (`^Y`).

Other constants are surrounded by single quote marks. `OpenQuote` is a constant defined as the double-quote character (it is enclosed in single quotes). `OneSpace` is a single space, and `Spaces` is equal to three spaces. `BlankLine` is a blank string (an empty line of text, consisting of only a Return).

The other constants include the actual typesetting codes, defined as strings (enclosed within single quotes).

Near the end of the constants list is `CloseQuote`, which is defined as two single quotes (apostrophes); however, since they cannot be enclosed in single quotes, we used ASCII decimal code numbers prefixed with the `#` character (see ASCII chart, issue #8; also found in many books and manuals).

At the end of the constants list are two *boolean* constants, designed so that Pascal statements using them would be easier to read. The constant `On` is a substitute for the Pascal condition `FALSE` (which is a reserved word in Pascal, and cannot be used to mean anything else). The word `Off` is a substitute for `TRUE`.

```
PROGRAM TypeWs (Input,Output);
CONST
    PItal      = ^Y;  {* These are the WordStar print *}
    PBold      = ^B;  {*  control codes used in text. *}
    PSpec1     = ^D;  {* You can change them to any single- *}
    PSpec2     = ^S;  {* character control codes. *}
    OpenQuote  = '"';
    OneSpace   = ' ';
    Spaces     = '   ';
    BlankLine  = '';       {* Below are typesetting controls: *}
    MergeCode  = '{mc}'; {* to control paragraph indents, *}
    QuadLeft   = '{ql}';  {* to end single lines, *}
    Italics    = '{cf3}';  {* Roman Italic Font 3, *}
    Bold       = '{cf4}';      {* Roman Bold Font 4, *}
    Special1   = '{cf397}'; {* Special Font 397, *}
    Special2   = '{cf396}'; {* Special Font 396, *}
    Regular    = '{cf2,10,12}';  {* Roman Font 2 size 10/12 *}
    CloseQuote = #39#39;
    on         = FALSE;
    off        = TRUE;
TYPE
    LineOfText = STRING[255];    {* Maximum string length. *}
VAR
    InFile, OutFile    : TEXT;    {* Straight text files.*}
    LineIn, LineOut    : LineOfText;
    n                  : INTEGER; {* A simple character counter.*}
    it                 : CHAR;    {* This is the char to test. *}
    NextOne, NextQuot  : BOOLEAN; {* The assumption conditions. *}
PROCEDURE OpenFiles;
VAR
    filename1, filename2 : STRING[14];
BEGIN
    WRITE('Type name of text file to typeset: '); READLN(filename1);
    WRITE('Type name of TYP file to create: '); READLN(filename2);
    ASSIGN(Infile,filename1);
    ASSIGN(Outfile,filename2);
    RESET(Infile);
    REWRITE(Outfile);
END;
PROCEDURE SwitchCodes;
BEGIN
  IF NextOne = on THEN
    BEGIN
      CASE it OF
          PItal : LineOut := CONCAT(LineOut + Italics);
          PBold : LineOut := CONCAT(LineOut + Bold);
          PSpec1: LineOut := CONCAT(LineOut + Special1);
          PSpec2: LineOut := CONCAT(LineOut + Special2);
      END;
      NextOne := off;
    END
  ELSE
    BEGIN
        LineOut := CONCAT(LineOut + Regular);
        NextOne := on;
    END;
END;
PROCEDURE FixQuotes;
BEGIN
    IF NextQuot = on THEN
      BEGIN
          LineOut := CONCAT(LineOut + OpenQuote);
          NextQuot := off;
      END
    ELSE
      BEGIN
          LineOut := CONCAT(LineOut + CloseQuote);
          NextQuot := on;
      END;
END;
PROCEDURE ReadSpaces;
VAR
    nextchars : STRING[3];
BEGIN
    nextchars := COPY(LineIn,n+1,3);
    IF nextchars = Spaces THEN
        BEGIN
          LineOut := CONCAT(LineOut + MergeCode + OneSpace + OneSpace);
          n := n+4; {* advance four characters to skip spaces *}
        END
    ELSE BEGIN
          LineOut := CONCAT(LineOut + it);
          n := n+1; {* advance one to skip space *}
        END;
END;
{* Start of the main section. *}
BEGIN
    OpenFiles;
    NextOne := on;
    NextQuot := on;
    LineOut := BlankLine;
    WHILE NOT EOF(InFile) DO
      BEGIN
        READLN(InFile,LineIn);
        n := 1;
        IF LineIn <> BlankLine THEN
          BEGIN
            it := LineIn[n];
            IF it = OneSpace THEN ReadSpaces;
            WHILE n <= Length(LineIn) DO
              BEGIN
                it := LineIn[n];
                IF it IN [PItal,PBold,PSpec1,PSpec2,OpenQuote]
                     THEN BEGIN
                            IF it = OpenQuote THEN FixQuotes
                            ELSE SwitchCodes;
                          END
                     ELSE LineOut := CONCAT(LineOut + it);
                n := n+1 {* ordinary advance one char *}
              END;
          END
        ELSE LineOut := QuadLeft;   {* Else line is blank *}
        WRITELN(LineOut); WRITELN(OutFile,LineOut);
        LineOut := BlankLine;
      END;
CLOSE(InFile); CLOSE(OutFile);
END.
```

Figure 4. *The Pascal program that substitutes typesetting codes for WordStar printing codes.*

The next section of the program defines the *type* of variables and the variables themselves. Pascal has rigid variable types that keep the programmer from making mistakes in the algorithm and keep the data entry typist from entering bad data.

Pascal's predefined types used in this program include `TEXT` (file), `STRING` (array of characters), `CHAR` (one character), `BOOLEAN` (logical condition) and `INTEGER` (whole number). You can define your own types; we defined one called `LineOfText`, which is a string of up to 255 characters.

We use `LineOfText` to define two variables: `LineIn` (read from the input file, `InFile`) and `LineOut` (written to the output file, `OutFile`). We also define a simple integer counter, `n`, to use as an index to the characters in the string array `LineOfText`.

We define the central character, `it`, as the one who gets checked for being a print code or an open quote. Finally we define the two *boolean* (true/false) conditions, `NextOne` (the next print code) and `NextQuot` (the next double-quote).

Procedures

A *structured* programming language is one that lets you define an entire set of operations under one "label" or "procedure" name. A procedure is similar to a BASIC subroutine, except that you use a procedure name rather than a subroutine line number when you execute the procedure. This makes it possible to make programs that are easy to read, because each set of operations has an easy-to-read name.

We define four procedures in our Turbo Pascal program: `OpenFiles`, `SwitchCodes`, `FixQuotes` and `ReadSpaces`. Their names give broad hints about their purposes. In a program, the names stand out as obvious procedures the program must perform in a logical order.

Why use procedures? When you are writing a complex program, you may find yourself repeating some instructions that could be consolidated into one procedure that can be tested and made to work properly. You could then rely on that procedure without having to re-invent it every time your program needed to perform that sequence of operations.

If you are writing several complex programs, you may find that some procedures can be made "common" to several different programs. For example, our `OpenFiles` procedure could be written to be useful for almost any program that has to open an input file and an output file.

Take a close look at the `OpenFiles` procedure. It starts with a `VAR` (variable) declaration that declares `filename1` and `filename2` to be 14-character strings. Although the program's main variables and constants were already defined, this procedure has its own variables that are exclusively used within the procedure. The variables `filename1` and `filename2` are used within the procedure to receive the names of files typed in during the program's run.

They are defined within the procedure so that (1) the variables are "hidden" in the procedure in which they are used, and (2) the variable names can be used in other procedures to hold other data.

Why are these two things important? Hiding variables within a procedure makes the procedure more universal, and therefore more useful if you intend to use the procedure in other programs. These "local" variables (ones defined and used only within a procedure) don't clutter up the variable section of the main program. Also, it frees the variable names so that they can be used for variables in other procedures or even in the main body of the program. This becomes important when you write many programs and borrow procedures from different programs.

The body of the `OpenFiles` procedure performs two `WRITE/READLN` statements to get the two filenames from the keyboard, then assigns these filenames to the file variables `InFile` and `OutFile`. The procedure then resets the `InFile` (moving the "file pointer" to the beginning of the file) and creates (or recreates) a new `OutFile`.

The `SwitchCodes` procedure starts with an `IF/THEN... ELSE` structure that selects a CASE structure if `NextOne` (the next print code) is `on`. The `CASE it OF` phrase sets up the CASE structure, which selects a different CONCAT operation for each print code. For example, when `it` is a `PItal` print code, the program executes the `CONCAT (LineOut + Italics)` function and assigns the result to `LineOut`. When `it` is a `PBold` print code, the program assigns the result of `CONCAT(LineOut + Bold)` to `LineOut`. In these case statements the former contents of `LineOut` is preserved and the new code is added to it.

We needed a program to quickly convert our WordStar-coded text into typesetting-coded text.

The procedure then sets the variable `NextOne` to `off` for the next time around.

If `NextOne` is `off`, the procedure skips the CASE structure and performs a CONCAT substituting the `Regular` typesetting code. It then changes `NextOne` to `on` for the next time around, so that the next substitution is an "on" code.

This section of the program can be expanded if you need to substitute different typesetting codes (we need only one typesetting code, `Regular`, to "turn off" all the other typefonts.

The `FixQuote` procedure works in a similar manner to flip-flop between one condition (`NextQuot` is `on`, or an open quote) and the other (`NextQuot` is `off`, or a closed quote). This causes the substitution of closed quotes for every second double-quote. Again, you could expand this procedure to detect special situations where you do not want the substitution of a closed quote for every second double-quote.

The `ReadSpaces` procedure starts with a definition of a local variable `nextchars` (a three-character string). The procedure uses Turbo Pascal's built-in COPY string function:

```
nextchars := COPY(LineIn,n+1,3);
```

In this example COPY copies three characters starting at `n+1` in the string `LineIn` to the string `nextchars`. The procedure tests these characters to see if they're spaces. If they are, the procedure substitutes a merge code and two spaces; if not, the procedure adds the character `n` (stored in `it`) to the line to be output. In both cases the procedure advances `n`, thereby skipping the space or spaces before moving on.

The Main Section

Every Pascal program has a *main section* that controls the operation of in a linear fashion; *i.e.*, step-by-step. The main section of our program starts by executing the procedure `OpenFiles` — the name by itself, followed by a semi-colon, causes the procedure to execute.

The program then sets up, or *initializes* the "global" variables. `NextOne` and `NextQuot` are set to `on`, and `LineOut` is set to a `BlankLine`. This is an important step — without it, you would not know the initial values of these important variables.

The `WHILE NOT EOF(InFile) DO` phrase defines the beginning of a DO/WHILE block which will execute over and over until the program reaches `EOF(InFile)` (the end of the input file). The first thing this block does is read one line of text and set the character counter `n` to one.

If the input line is *not* (`<>`) a blank line, the program assigns the variable `it` to the `n`th character (in this case, the first character) of `LineIn`. The first test of `it` is whether `it` is a space (`OneSpace`); if so, the program executes the `ReadSpaces` procedure.

Whether or not the line began with a space (and whether or not the program executed `ReadSpaces`), the program moves on to process the characters of `LineIn`. The process is set up as another DO/WHILE block; this one executes until the counter `n` reaches the end of `LineIn`:

```
WHILE n <= Length(LineIn) DO
   BEGIN
```

The DO/WHILE block uses two nested IF/THEN/ELSE statements: the first one tests to see if `it` is in the set of print codes (`PItal`, `PBold`, and so on), and the second tests to see if `it` is an open quote (double-quote is the same as an open quote). If `it` fails either test, `it` is simply concatenated to the output line. If `it` passes the first but not the second, the program executes `SwitchCodes`; if `it` passes both tests the program executes `FixQuotes`.

At the end of these operations, the program adds one to the character counter `n`, and the DO/WHILE block runs again, unless it is stopped by the WHILE statement because the counter is pointing to the end of the input file.

The original IF/THEN statement framing this DO/WHILE block is the one that checks if `LineIn` is not a blank line. If `LineIn` *is* a blank line, the program skips the entire block and assigns a `QuadLeft` to `LineOut`.

Finally the program writes `LineOut` to the screen and to the output file with two `WRITELN` statements. The last steps are to assign a blank line to `LineOut` and to close the files.

End Of Lesson

The next installment of this Turbo Pascal tutorial will look at math functions, including writing your own functions, and other aspects of structured programming. Until then, practice using Turbo Pascal to process text files and strings.

You can add code to our program example to handle all the WordStar printing codes; you can even use this program to substitute another word processor's codes for WordStar codes, and thereby transfer text files from WordStar format to another word processor. Simple modifications (changing the constants from typesetting codes to another word processor's printing codes) would do the trick.

NEWWORD®
VERSION 2
NOW ONLY $100!

HERE ARE SOME OF THE FEATURES YOU GET:
- 30 day money back guarantee
- WordStar/MailMerge compatible — file, keystroke, command
- Mergeprint is standard — including conditionals (if, then, else)
- Displays underline, bold, & inverse on the screen*
- Unerase — delete a line by accident? Unerase it!
- Microjustified printing on dot-matrix printers
- Word Plus™ spelling checker $50 extra

MONEY BACK GUARANTEE
ORDER NOW!

(800) 932-2526 • (415) 932-2526 (CA only)
OR SEND CHECK OR MONEY ORDER TO:

Newstar Software, Inc.
1601 Oak Park Blvd.
Pleasant Hill, CA 94523

- send $100 for Newword
- $3.00 shipping & handling
- CA residents add sales tax
- VISA/MC accepted

NEWSTAR

*terminals with non-embedded attributes only
offer applies to CP/M™ versions only
Newword, CP/M, WordStar, MailMerge are trademarks of
Newstar Software, Digital Research, and MicroPro
Word Plus is a trademark of Oasis Systems

Circle number 9 on the reader service card.

ADVANCED USER/Bruce Morgen

The Advanced User In the 8-bit World

The first installment of this new column describes the wonders of ZCPR (a system enhancer) and LRUN (running programs in libraries).

TRY TO UNDERSTAND.... EVERYBODY'S GOING MICRO IT'S NOTHING PERSONAL, IT'S BUSINESS
© L.Shoemaker 85

This column is based on a belief I have that there are a vast number of computer users — particularly in the CP/M-compatible 8-bit world — that are neither beginners nor programmers.

We are experienced CP/Mers who understand how to use the free software in the public domain, and how to modify commercial packages to get the most out of our computer investments. We are constantly on the lookout for new techniques to improve system performance, but we rarely resort to writing our own code — with the huge treasure trove that is the CP/M-compatible public domain, why reinvent the wheel? It is to this group of knowledgeable non-programmers — advanced users, if you will — that this space is hereby dedicated.

Being a charter member of this category, I'll be writing about some of my own discoveries, but I hope that A.U.C. (Advanced User Column) will grow into an interactive sort of creature, becoming increasingly vital and creative as more readers share their own goodies with the rest of us. So, if you already understand the basics of CP/M and if the term "XMODEM" doesn't conjure up visions of explicit lewdness at 1200 baud (or even if it does), read on.

ZCPR — To 3 or Not To 3

I may be opening up a can of worms here, but it's getting harder and harder to understand why some CP/Mers are so skeptical about ZCPR3 (*for a review of ZCPR3, see issue #11/12*). The rationales I've heard range from "Who needs the complications" to "ZCPR2 (or 1) is good enough, why change?" to "if it's not real CP/M, I don't want to know about it."

Let's examine these assumptions one at a time.

First of all, it is true that installing ZCPR3 from scratch on new hardware requires some knowledge of assembly language and the computer involved as well as a good deal of hexadecimal arithmetic; however, there are turn-key packages already developed for most of the more popular CP/M machines that can be up and running in less than an hour.

Good implementations are available for post-1983 Kaypros, the Morrow Micro-Decision series, the Xerox 820-II and the Osborne I, among others. These are free for the downloading on many of the better RCP/M systems, or you can get them from the main ZCPR3 BBS, Z-Node Central, at 415-489-9005. Sysop David McCord is a cooperative person with lots of answers, and the board is frequented by ZCPR3 author Rick Conn, as well experts like programmer Joe Wright and Frank Gaude' (president of Echelon).

Echelon is a small, committed company formed to sell ZCPR3 and related software to users and OEMs (re-packagers of computer systems), but you don't have to be a customer to get the benefit of their expertise or to download the latest bug-fixed ZCPR3 programs. The company is Rick Conn's authorized agent for ZCPR3 sales and licensing, and has also done a very good job in originating, controlling and documenting the revision of the hundreds of programs involved. As Tony Bove and Cheryl Rhodes pointed out in their recent review (*issue 11/12*), the support of the software by Echelon is one of the major differences between "Z3" (as most of us call it) and previous ZCPR efforts.

As to how good Z3 is, you really have to live with a good implementation to appreciate it. Remember that Digital Research (makers of CP/M) never called CP/M an operating system (the 2.2 manual refers to it as "a monitor control program"). Although compatible with all properly-written CP/M programs, ZCPR3 is an operating system in the full sense of the word, with a massive set of utilities and options that rivals unwieldy minicomputer-derived systems like AT&T's UNIX and Digital Equipment's Micro-VMS. This is not just my opinion — Rick Conn uses minicomputers every day, and many Z3 proponents have similar experience.

On its own, each ZCPR3 feature and utility is impressive; viewed as the complete system it is, ZCPR3 is by far the best single-user operating environment for 8-bit computing. Its designed-in flexibility and nearly endless array of options and features are unlikely to be completely explored by any one user, but serve to assure that there will be an appropriate setup for anyone who wants to really take control of their computerized workplace. Despite having a (not-quite) 16-bit processor to work with, MS-DOS doesn't even come close.

As far as "brand-loyalty" to authentic CP/M goes, I tip my hat to Professor Kildall and colleagues at Digital Research for the great achievement CP/M — particularly the 2.2 version of the BDOS — was and is. With that out of the way, let me say that I don't think Digital Research has shown very much interest in the 8-bit community, at least not recently, so I'm not sure they deserve all that devotion. CP/M and its bundled utilities are like a good beginner's tool set. An intelligent, motivated user soon moves on to better things, and there is certainly no good reason to make the spartan CP/M CCP sacrosanct. In other words, get ZCPR3 — there is no other non-hardware change that will improve your 2.2-compatible system nearly as much.

Poke & Go

One of the great advantages of ZCPR3 is its dynamically loadable system segments, which include resident command packages (*.RCP), flow control packages (*.FCP), input/output packages (*.IOP) and named directories (*.NDR). Of the features these segments provide, only the named directories and I/O redirection were available under ZCPR2.

The RCP concept, in particular, puts a lot of new power in the hands of the advanced user. By putting commands like ERA, REN, CP (a handy PIP-like file copy command) and TYPE (the Z3 version has pagination and wildcard capability) along with new features like P (peek at memory) and POKE (change memory) in an RCP, the CCP-resident commands can be restricted to those that cannot be implemented any other way. This keeps the Z3 CPR (Command Processor Replacement) down to a CP/M-compatible 2K. Commands that must be CPR-resident include an enhanced SAVE and two powerful features unavailable in "vanilla" CP/M — GET and GO.

GET loads a file into memory starting at a specified address — typing `GET 100 WS.COM↵` loads WordStar into the normal TPA (transient program area), but does not run it (the TPA starts at 100H in most Z3 systems, which is compatible with CP/M 2.2). A subsequent GO command can then run the program. When combined with an RCP-resident POKE command — which executes user-defined memory alterations without corrupting the TPA — virtually any program can be temporarily patched at run time: just GET the program into memory starting at 100H, POKE the locations you want to change and then GO, executing the altered program.

Putting this power to work in the real world brings in another ZCPR3 goody, the ALIAS program. ALIAS creates a tiny (1K) ".COM" file that inserts a command line - the commands are separated by semicolons — into ZCPR3's multiple command line buffer, from which the commands are passed to ZCPR3 for sequential execution.

Using WordStar as an example, let's suppose your WS.COM is installed to come up in non-document mode when a file name is included on the command line, but you'd like it to come up this time in document mode. The following command line will do the trick:

```
GET 100 WS.COM;POKE 0378 00;GO $1↵
```

Just "build" an ALIAS called WSD.COM with that "script", and when you type —

```
A>WSD TESTWSD.TXT↵
```

— WordStar will come up with the file in document mode. In case you haven't figured it out, the **$1** in the ALIAS script is a SUBMIT-style passed command line parameter.

WSD.COM is a very simplified example of a "POKE&GO" ALIAS. Just looking at WordStar applications alone, you could have, for example, a half-dozen or more custom printer patches available on demand, all derived at run time from a single copy of WordStar at a disk space cost of 1 to 8 kilobytes (depending on how your system allocates disk space) per patch. Sure beats all those 18 kilobyte WordStar ".COM" files that some people put up with to accomplish the same thing!

This is just a hint at the power and convenience that ZCPR3 brings to everyday computer use. By the way, if your Z3 implementation doesn't include the GET command, you can still use POKE&GO with some help from ZEX, the souped-up memory-based ZCPR3 enhancement of the CP/M SUBMIT and XSUB programs. Here's a simplified ALIAS command line script:

```
ZEX WSGET;POKE 0378 00;GO $1
```

You'll also need a file called WSGET.ZEX that looks like this:

```
WS
^.
^#
X
```

ZEX will run and immediately exit WordStar without leaving any "footprints". This is the functional equivalent of the GET command, although somewhat slower. Only one WSGET.ZEX is required to support any number of POKE&GO ALIASes.

The joys of LRUN

Most CP/Mers are aware of ".LBR" files from their RCP/M experiences. Gary Novosielski's brainchild is a splendid way to organize public domain (free) software packages for subsequent downloading, but it has other virtues as well. One of the most useful is made possible by a handy little (2K) program called LRUN, also a Novosielski creation (super-programmer Ron Fowler contributed a couple of key LRUN routines).

If you use a lot of small utilities — what advanced user doesn't? — you quickly find your disk space being filled up in 2K-to-8K chunks by programs that often have only 1K or less of code. On a floppy-based computer the problem is two-pronged because it's so easy to come up against CP/M's filenames-per-disk restriction, which is usually 64 files or less. The code for 350 or more 1K ".COM" files would fit handily on a double-sided double-density floppy, but CP/M won't let you do that because there isn't enough space on the directory tracks!

The problem is a bit subtler on hard-disk systems where disk space isn't quite so dear, but their larger minimum-space-per-file allocations, usually 4K or more, are still a pointless space-waster when occupied by a sub-4K ".COM" file. Within an ".LBR" file, however, member files can be as short as 128 bytes (one CP/M record, a half-page of memory space).

Enter LRUN — just put your little gems in a library file called COMMAND.LBR, type `LRUN command`↵ and most ".COM" files will run just like they did outside COMMAND.LBR. Actually, the LRUN syntax is a bit more flexible than that. Here's what LRUN's built-in help screen has to say:

```
LRUN Ver 2.2
Correct syntax is:

        LRUN [-<lbrname>] <command line>

Where <lbrname> is the optional library name
(Note the preceding ''-''. ) If omitted,
the default command library is used.

<command line> is the name and parameters
of the command being run from the library,
just as if a separate .COM file were being run.
```

COMMAND.LBR is the default command library. Of course you can rename LRUN.COM to anything you want to shorten the necessary command line.

LRUN version 2.0 is the most widely available variant and it works well unless you use user areas and want to access the programs in COMMAND.LBR from any of them — version 2.0 expects to find COMMAND.LBR in the currently logged user area. Versions 2.1 and 2.2 allow the assembly-time option of specifying an alternate drive and user where LRUN will look for COMMAND.LBR. LRUN22.COM can also be patched with a debugger or hex editor to change the alternate drive and user — this is quicker than reassembly and the only way to do it if you don't own a macro-assembler.

For optimum use of LRUN, ZCPR2 and ZCPR3 users can enable an assembly-time option in their CPRs called CMDRUN and rename LRUN.COM to CMDRUN.COM. This allows you to run any program in COMMAND.LBR with the same syntax you would use if were a standalone ".COM" file! There are special LRUN variants, LRUNZ.COM and LRUNZ302.COM, for ZCPR2 and ZCPR3 respectively. LRUNZ will run fastest if installed (using the ZCPR2 GENINS utility) with internal path enabled and the first path element set to the drive and user area where COMMAND.LBR resides. LRUNZ302 installs automatically with the ZCPR3 Z3INS utility, but is noticeably slower than LRUNZ because it can only follow ZCPR3's external path, which is usually at least 3 elements long.

ZCPR or no ZCPR, I've found LRUN22 to be the fastest of all because its mini-path (only two elements) is straight in-line code. For best overall execution speed, set the ZCPR equates CMDRUN and ROOTONLY to TRUE and put both your chosen LRUN (as CMDRUN.COM) and COMMAND.LBR at the end of your command search path, the ZCPR "ROOT" directory.

If using LRUN seems a bit on the complicated side, don't let it intimidate you. The rewards are too great to pass up, particularly for floppy-based systems. The disk I'm using right now has well over a hundred executable programs on it, with 16K and a dozen directory entries still available. It's well worth the (minor, if any) trouble, to say the least.

Access to non-".COM"s

One thing LRUN will not do for you is allow access to non-executable files within an ".LBR" file. Suppose you have a substantial number of phrases, short paragraphs or what have you that you'd like to be able to "read in" to text files you are creating — using the WordStar Control-`KR` command, for example. These small files run into the same problems that LRUN solves for ".COM" files: disk space allocation and directory space. The popular library programs like LU310, NULU and LSWEEP13 won't help here, but a clever utility called LBRDISK written by Jim Lopushinsky will do that very non-trivial task.

LBRDISK — the current revision level for the CP/M 2.2-compatible version is 2.21, distributed as LBRDSK23.LBR — borrows a CP/M Plus concept in that it is a resident command extension (RSX). An RSX runs in high memory, just under the CCP or CPR, rather than in the standard TPA. Because CP/M 2.2 was not written with RSXs in mind, the first step in running LBRDISK is to run an RSX manager program called SETRSX.COM, which is supplied in LBRDSK23.LBR. SETRSX intercepts console and program warm boot requests, preventing any RSX modules loaded below it from being destroyed. Running SETRSX a second time will remove the manager and any RSX modules residing below it.

LBRDISK itself is an RSX that fools BDOS into believing that the member files in an ".LBR" file are standalone files in a virtual disk directory — a somewhat difficult concept that the following terminal session might render more understandable:

```
A>SETRSX↵
```

This program loads the RSX manager.

```
A>LBRDISK C0:=A13:MEX↵
```

This command line creates the virtual disk directory — drive C, user 0 — from the member files of MEX.LBR on drive A, user 13. You can move to "drive" C and check its directory:

```
C>DIR↵
          Drive C0 files: 24 space used: 48k (12K FREE)
0        .MEX  2K ¦ GETLBR   .MEX   2K ¦ KPCURA   .MEX  2K ...
BELCAL   .MEX  2K ¦ GETLIBYE.MEX   2K ¦ L1       .MEX  2K ...
CL       .MEX  2K ¦ IMHERE   .MEX   2K ¦ L2       .MEX  2K ...
CURA     .MEX  2K ¦ INI      .MEX   2K ¦ MAP      .MEX  2K ...
GET      .MEX  2K ¦ INT      .MEX   2K ¦ NAMEOFF  .MEX  2K ...
GETBYE   .MEX  2K ¦ KEYS     .KEY   2K ¦ PHONE    .PHN  2K ...
```

To remove the RSX manager:

```
C>A:SETRSX↵
RSX Removed
Warm Boot

A>
```

Just how much space can you save? Well, the MEX.LBR in the example is actually a 14K file — a savings of 34K less 2K for SETRSX.COM and 8K for LBRDISK.COM.

Like all high-memory-resident programs — BYE, EX14 and most screen dump utilities are also in that category — LBRDISK does cut down on available TPA and can put a minor but noticeable crimp in the performance of some larger programs by constricting buffer space. It only allows one file on the virtual drive to be open, so altering the files with WordStar or Perfect Writer is a no-no.

Even when a small editor like ED.COM or VDO is used, caution is in order — keep in mind that ".LBR" files are sequential and any lengthening of an existing member file can trash the next file in the sequence. Like NULU10, LBRDISK will set all internal ".LBR" CRCs to zero if a write operation occurs, so it might be prudent to patch LBRDISK to read-only and do file alterations conventionally. Patching instructions are provided in LBRDSK23.LBR.

NULU news

Flash!: the latest version of NULU has LU-style CRC capability and a bunch of nice new refinements, including a terminal patch overlay. Some very subtle bugs may or may not still be in the program, but I'm convinced it's 99+% perfect. Look for NULU12.LBR on an RCP/M near you, or get NULU11.LBR and patch the ".COM" file with NULU11F1.ASM, a minor bug fix patch. Kudos to author Martin Murray for one heck of a versatile utility and for keeping his creation down to only 15K!

Speaking of Texans, Seneca RTDS (Remote TurboDOS System) sysop Sigi Kluger, A.K.A. ESKAY, has written a 2K program that computes both CHEK and CRCK values with full wildcard capability. Download NCRCK1 (".OBJ" or ".LBR") for another winner from the author of TYPEL36 and the amazing FINDU11 (more on these in later editions of this column).

Coming Attractions

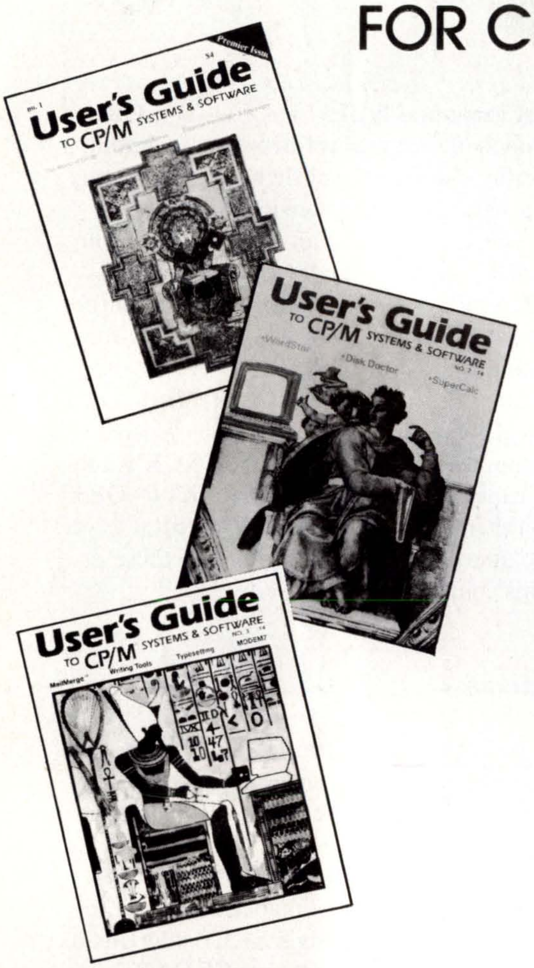
I'm coming up fast on my self-imposed 3,000 word limit for the inaugural episode of A.U.C., so let me preview the next installment (barring new and better ideas from you, dear reader). We'll take a look at the Plu*Perfect Systems "Public" patch (yes, it's free - thanks guys!) for the 2.2 BDOS, which is my favorite CP/M enhancement next to ZCPR3; and I'll talk about Tony Fleig's XCCP and Dennis Hamilton's EZCPR, either of which allows CP/M 2.2 holdouts to sample many of the advantages of ZCPR without any installation at all, and take a brief tour of Ron Fowler's MEX, one of the best 8-bit communications packages ever, bar none, and possibly the finest CP/M application ever written that's free for the downloading. I'll also try to overcome my reticence about expressing my opinions.

In the meantime, let me know if I'm providing too little or too much detail so far. For example, I didn't describe how to find which WordStar addresses to POKE to support the POKE&GO technique, operating on the assumption that A.U.C. readers have already met up with and used DIFCOM16 and/or the ZCPR2/3 DIFF programs. Please tell me whether I guessed right, OK?

```
A0:SYSTEM>BYE↵
```

BACK ISSUES

A TRAINING LIBRARY FOR CP/M & MS-DOS USERS



User's Guide TO CP/M SYSTEMS & SOFTWARE
Premier Issue
User's Guide TO CP/M SYSTEMS & SOFTWARE
WordStar
Disk Doctor
SuperCalc
User's Guide TO CP/M SYSTEMS & SOFTWARE
MailMerge
Writing Tools
Typesetting
MODEM7

User's Guide TO CP/M SYSTEMS & SOFTWARE
CP/M Plus
SuperCalc
Disk Utility
dBASE II
User's Guide TO SYSTEMS & SOFTWARE
CP/M Plus Tutorial
dBASE II Tutorial
Perfect Filer

#1R: CP/M® (revised)

Learning to Use CP/M • Impatient User's Guide® to CP/M • Using CompuServe® • Preventing Computer Disasters •

#2: WordStar® & SuperCalc®

A Guide to Using WordStar • Impatient User's Guide to WordStar • WordStar Installation Guide • A Guide to Using SuperCalc • Disk Doctor: The Rx For Bad Disks • For Starters: Word Processing & Spreadsheets •

#3: MailMerge & Modem7

MailMerge: WordStar's Printing Robot • Impatient User's Guide to MailMerge • Using DocuMate Plus and The Word Plus • Using Footnote and Bibliography • Modem7 Tutorial and RCP/M List • WordStar File and Print Summary • For Starters: Word Processing Power •

#4: CP/M Plus

Overview of CP/M Plus • Guide to SuperCalc (Pt. 2) • SuperCalc SuperSummary • CP/M Plus Operating Tools • Reference Guide to DU (Disk Utility) • Overview of Floppy Disks • System Overview: CompuPro 8:16 • For Starters: Exploring dBASE II •

#5: dBASE II

Introduction to dBASE II • dBASE II Application: Subscription Fulfillment • Perfect Filer Overview & Starter • Using CP/M Plus (Pt. 1) • CP/M Plus: Inner Workings • Hard Disks For Small Computers • WordStar on the TeleVideo 950 • System Overviews: Radio Shack Model 100, PMC MicroMate •

Special Offer:

#1-#10 for $30, #7-#13 for $20, #1-15 for $50. (cover price: $4.50 and up)

☐ SPECIAL OFFER: Issues #1-#10 for $30	(+$10 First Class)	$________
☐ SPECIAL OFFER: Issues #7-#13 for $20	(+$5 First Class)	$________
☐ SPECIAL OFFER: Issues #1-#15 for $50	(+$15 First Class)	$________

Payment must accompany order. TOTAL ENCLOSED:
We accept Visa and Mastercard. Phone: 415-364-0108.

Name ______________________________

Company ______________ Title ______________

Address ______________________________

City, State, Zip ______________________________

☐ Check if you are already a subscriber, or are sending in a subscription card.

Make checks payable to: User's Guide. Offer expires 12/31/85
Back Issues Dept.: P.O. Box 5245, Redwood City, CA 94063.
Please allow 6–8 weeks for processing
or add First Class postage.

User's Guide is a trademark of TUG Inc.
CP/M is a registered trademark of Digital Research, Inc.
MS-DOS is a trademark of Microsoft.

UG15b

#6: Communications

CompuServe Electronic Mail • CompuServe Public Access & File Transfers • Communications Programs: COMMX and MITE • Guide to Using WordStar (Pt. 2) • Converting From Perfect Writer to WordStar • WordStar vs. Perfect Writer • CP+ and Organizer: Menus for CP/M • Cache/Q: A System Enhancer • Glossary of Computer Communications Terms • For Starters: Communications •

#7: T/Maker III & CP/M Plus

T/Maker III: Integrated Text Processor, Spreadsheet and Table Maker • Using CP/M Plus, Pt. 2: Utilities For Automation • FILEBASE for Personal Computers • Proportional Printing With Perfect Writer • MS-DOS and PC-DOS Overview, Pt. 1 • Making Communications Connections • For Starters: Brief Tour of CP/M Systems • System Overview: DEC Rainbow • **Annual Index for issues 1-6** •

#8: Accounting Packages Compared

Show Report: Softcon and 9th West Coast Computer Faire • For Starters: Computerized Accounting • Small Business Bookkeeping: Taranto's General Ledger • MS-DOS and PC-DOS, Part 2 • Comparing Accounting Packages: Accounting Plus vs. SBT • Using MiniLedger • Patching With DDT and SAVE • dBASE II Tutorial, Part 2: Indexing and Sorting • ASCII Chart •

#9: Multi-User and Concurrent Systems

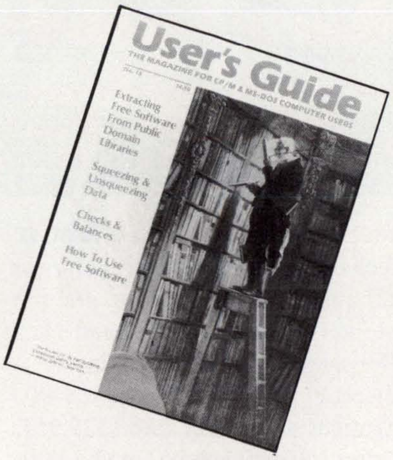
For Starters: Multiple Users and Multiple Computers • The Sleaze Patrol • The Meaning of Concurrency • Concurrent CP/M-86 and MP/M-86: Files, Disks, Spooling and Printing • Concurrent CP/M-86: Versions for DEC Rainbow, IBM PC • MP/M-86: Version for CompuPro • NewWord Evaluation • Bible Study by Computer • Punctuation & Style Evaluation • System Overviews: OSM Zeus 4 Network, Ampro Bookshelf Computer •

#10: Amateur Programming

For Starters: Computer Languages • File Handling Utilities • Running MBASIC Programs • Turbo Pascal • From Source To Object: Executable Programs • MS-DOS and PC-DOS (Part 3) • Batch Processing in CP/M and MS-DOS • The CBASIC Compiler: CB-80 and CB-86 • Giving Data to a BASIC Program From the System Command Line •

#11/12: Special Double Issue

For Starters: Free Software: Getting It • The Sleaze Patrol • Columns By WordStar • What Is ZCPR3? • CLIP: a Command Line Interpreter • Comparison: EUREKA! vs. CATALOG • Calling Bulletin Boards For Free Software • Using Modem7 • Free Software: the Newest Modem7 (MDM740) • WordStar Games Your Mother Never Taught You (Part 1) • Make NewWord Work For You • Phone List of RCP/M Bulletin Board Systems • CP/M Error Message Index •


User's Guide
THE MAGAZINE FOR CP/M & MS-DOS COMPUTER USERS
Extracting Free Software From Public Domain Libraries
Squeezing & Unsqueezing Data
Checks & Balances
How To Use Free Software

#13: Using Free Software

For Starters: Free Software: How To Use What You Get • ESCapades • Squeezing and Unsqueezing Files • Library Files and Utilities • Free Software: Cataloguing Your Disks • Evaluation: Checks & Balances • Application: Local Network of Bowling Centers • **Annual index for issues 7-12** •

Act now to complete your set of *User's Guide* back issues! Limited offer while supplies last! This may be your last chance to get a complete set!

User's Guide

P.O. Box 5245
Redwood City, CA 94063

BACK ISSUES

Order now while supplies last! Payment must accompany order:

#1, 2, 3 or 4 Qty. _______ at **$6** each = $_______________
#5, 6, 7, 8, 9 or 10 Qty. _______ at **$5** each = $_______________
#11/12 Qty. _______ at **$8** each = $_______________
#13 or 14 Qty. _______ at **$4.50** each = $_______________
CP/M Summary Cards Qty. _______ at **$2** each = $_______________
First Class Postage and Handling. Qty. _______ at **$1.50** each **OR** **$1** each for 4 or more. $_______________
Foreign airmail add **$3** per copy or **$10** per 4 copies = $_______
TOTAL ORDER: $_______________

Phone: 415-364-0108.

Payment by charge card: ☐ Visa ☐ Mastercard Exp. date _______
Card # _______________ Interbank # _______
Signature required: _______________
Name _______________
Company _______________ Title _______
Address _______________
City, State, Zip _______________
☐ Check if you do not want promotional mailings.

Make checks payable to ***User's Guide***. Offer expires 12/31/85
Back Issues Dept.: P.O. Box 5245, Redwood City, CA 94063.

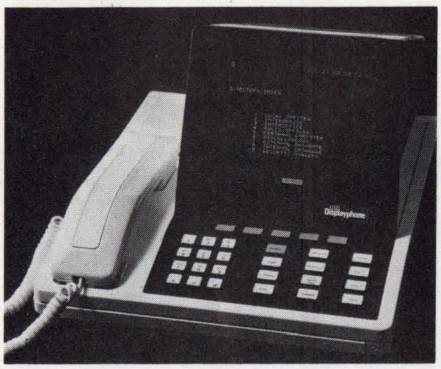
Please allow 6–8 weeks for processing
or add First Class postage.

UG15bi

New Products *continued from page 57*

built-in modem. The newest Displayphone PLUS features an internal 1200 baud (212a-compatible) auto-answer modem, and full DEC VT100 and VT52 terminal emulation. IBM 3101 and ANSI 3.64 cursor addressing are supported, plus a fifth cursor addressing protocol allows the Displayphone PLUS to emulate terminals manufactured by Data General, ADDS, Hazeltine, Televideo and others.

Manufactured in Nashville (TN), with a suggested retail price of $1595, the Displayphone PLUS also features a full-sized retractable keyboard, amber screen, parallel and RS-232C serial interfaces and telephone unit with a 90-number directory, auto-dial, and hands-free speaker feature. The telephone unit can be optionally adapted for use in a key telephone system. Standard operation is via modular wiring to accept two standard central office lines.



Northern Telecom Inc., 565 Marriott Drive Suite 300, Nashville, TN 37210. (615) 885-3510.

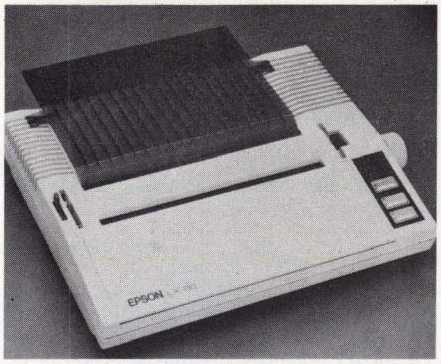
Circle 57 — reader service card.

Printers

Epson Dot Matrix News

Epson America (Torrance, CA) announces new 80-column dot matrix printer models.

The dual mode Spectrum LX-80 offers 160+ typestyles, selectable by resetting DIP switches on the printer, or by programming the bidirectional printer. Draft mode for the LX-80 is 100 characters per second (cps), near letter quality mode is 16 cps. and parallel interface, friction feed and 1K of buffer memory are included in the $349 retail price of the basic LX-80.



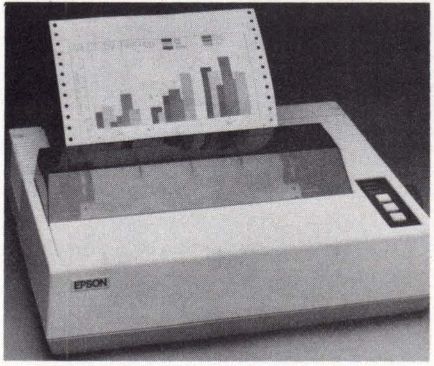
Options are tractor feed ($39.95), cut sheet feeder ($99.95), and 32K buffer board ($175).

Circle 58 — reader service card.

Rainbow Printer

The Epson JX-80 offers seven-color printing using a four-color ribbon, as well as single-color printing using a black ribbon, with a 160 cps print speed. The 128 standard typestyles are software-selectable as well as switch selectable using the printer panel control switches. Additionally, you can download special fonts you design into the JX-80's memory.

The JX-80's dot addressable graphics has the same vertical and horizontal scale (one-to-one), allowing true circles to be produced. Characters have full descenders (9 by 9 dot-matrix), and feature proportional spacing. A standard parallel interface makes the JX-80 software-compatible with the FX-80. Also included in the $800 retail price are both tractor and friction feeds.



Optional serial or IEEE 488 interfaces can be added using Epson's standard interface board options.

The JX-80 will be available in September at retail stores. Lotus, VisiCorp, Sorcim and Koala Technologies are among companies presently developing software for the JX-80.

Circle 59 — reader service card.

Accessories

Z80 Adapter and SCSI Expansion Bus

Ampro Computers Inc. announces SCSI/PLUS Z80 Adapter. SCSI/PLUS is a daughter board that plugs into the Z80 CPU socket of a Z80 based host computer. The Z80 Adapter's SCSI/PLUS bus interface meets the specifications for the popular SCSI (Small Computer System Interface — formerly called "SASI"), providing a general purpose I/O expansion bus.

SCSI/PLUS fully supports all SCSI Initiator and Target functions, including bus arbitration and disconnect/reselect. In addition, Ampro's SCSI/PLUS Initiator function extension allows connection of up to 64 SCSI/PLUS Target devices, rather than the usual eight-device limit of SCSI.

Applications include both direct and shared use of a variety of controllers and devices. For example, an SCSI Winchester hard disk controller, and modules providing clock/calendar, serial port expansion, RAM disk, and one or more Z80 hosts can be present on the same SCSI/PLUS bus.

The Z80 adapter adds "expandability" to closed Z80 host systems, allowing network interfaces, hard disks, etc., yet requires only 250mA of power (exclusive of Z80 CPU removed from host), which it obtains from the host CPU socket.

AMPRO Computers, Inc., 67 East Evelyn Ave., Mountain View, CA 94039. (415) 962-0230.

Circle 64 — reader service card.

International Trends in Manufacturing Technology
Robot Vision
Alan Pugh
Editor Professor Alan Pugh
Artificial Intelligence
The Search for the Perfect Machine
Lawrence Stevens
Art and Computers
The First Artificial Intelligence Coloring Book
Harold Cohen, Becky Cohen, Penny Nii
Foreword by Edward Feigenbaum
The Super-Intelligent Machine
Adrian
The Fifth Generation

Paul Winternitz

BOOKS/John Barry

AI Books Thrive in PA

While many people are ready to write epitaphs for computer-book publishing, one book genre seems to be alive and thriving: artificial-intelligence and robotics books. A recent display at Stacey's Bookstore in Palo Alto, California, featured more than 20 titles dedicated to AI and robotics.

The books in the display ranged from the paperback edition of the classic *The Fifth Generation* by Edward Feigenbaum and Pamela McCorduck to mind-numbing textbooks published several years ago. Stacey's specializes in technology books, and Palo Alto is in Silicon Valley, so this large selection of AI/robotics books may not be representative of what all stores stock. Still, the quantity of books on the subject was impressive, particularly for a discipline that's reputed to be rather obscure.

Copyright © 1985 John Barry

Here's the majority of the books in that display.

- *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World* by Edward Feigenbaum and Pamela McCorduck. Signet (published by New American Library), 1984. $3.95; 334 pages.

Addison Wesley first published this book in hardcover. Signet's softcover version is revised and updated to keep up with changes in technology. Feigenbaum is a guru and founding father of artificial intelligence; his coauthor is a novelist and science writer. Their contention is that the Japanese will dominate artificial intelligence unless America responds to their challenge.

- *The Super-Intelligent Machine* by Adrian Berry. Jonathan Cape Ltd., London, 1983. $14.95; 182 pages. Hardcover.

English author Adrian Berry traces the development of artificial-intelligence research and speculates on what the future holds. He touches, for example, on the work of Roger Schank, author of *The Cognitive Computer* (see *User's Guide* #14, "1984 in Review").

- *Into the Heart of the Mind* by Frank Rose (also see *User's Guide* #14).

- *Artificial Intelligence: Tools, Techniques, and Applications*, edited by by Tim O'Shea and Marc Eisenstadt. Harper & Row, New York, 1984. $28.50; 497 pages. Paperback.

This anthology features 15 essays on artificial intelligence by "established experts in the field." Part I, Tools, contains chapters such as "Advanced LISP programming." Part II, Techniques, tells readers "How to Get a Ph.D. in AI" and offers four other essays. Part III, Applications deals with "Computer Vision," "Industrial Robotics" and other topics.

- *Expert Systems: Principles and Case Studies*, edited by Richard Forsyth. Chapman and Hall, New York, 1984. $19.95; 231 pages. Paperback.

First published in England, this book is aimed at readers who already know about computing but are not up to date on the latest research into and applications of "expert" systems. The books defines *expert system* as "a software package which encapsulates specialist knowledge about a particular area of expertise and is capable of making intelligent decisions within that area." Some of the areas where real and working expert systems are now used, the book informs us, are medical diagnosis, geology, organic chemistry, and computer faultfinding. From the same publisher: *A Practical Guide to Designing Expert Systems* by S.M. Weiss and C.A. Kulikowski.

- *Artificial Intelligence: The Search for the Perfect Machine* by Lawrence Stevens. Hayden Book Company, Hasbrouck Heights, NJ, 1985. $14.95; 177 pages. Paperback.

This book attempts to cover virtually the entire realm of artificial intelligence—from an overview of the subject to biological/psychological/epistemological/ontological explorations of AI's attainability. In contrast to some of the dense tomes mentioned here, this book is mainly an appetite-whetter.

- *Principles of Artificial Intelligence* by Nils J. Nilsson. Tioga Publishing Company, Palo Alto, CA, 1980. $30; 476 pages. Hardcover.

The author of this book claims that previous books on artificial intelligence divide the subject into major areas of application, such as robotics, machine vision, and natural-language processing. Nilsson's goal, he writes, is to describe fundamental ideas that underline many of these applications. "The book is designed as a text for a senior or first-year graduate course in AI."

- *The Handbook of Artificial Intelligence*, edited by Avron Barr, Edward Feigenbaum, and Paul Cohen. William Kaufmann, Inc., Los Altos, CA (three-volume set; volume 1 copyright 1981) $39.50 per volume. Hardcover.

This massive three-volume set contains 200 articles on AI research from universities and laboratories nationwide. Conceived and produced at Stanford University, its producers hope it will become the standard reference work in the rapidly growing field of artificial intelligence. The series includes an extensive AI bibliography.

- *The Mind and the Machine: Philosophical Aspects of Artificial Intelligence*, edited by Steve Torrance. Ellis Horwood Ltd (a division of John Wiley & Sons, New York), 1984. $29.95; 213 pages. Hardcover.

Another English enterprise, this book is based on presentations at a weekend meeting in 1983 by philosophers, AI researchers, computer scientists, and workers in related fields. This book, which is part of a series on AI, also includes an extensive bibliography.

- *Artificial Intelligence Programming* by Eugene Charniak, Christopher Riesbeck, and Drew McDermott. Lawrence Erlbaum Associates, Hillsdale, NJ, 1980. $29.50; 323 pages. Hardcover.

According to this book, "The practice of AI is, of course, the writing or programs...almost all serious programming in artificial intelligence is done in the language LISP." As a result, the majority of the AI programming book is devoted to LISP. The book is intended primarily as a textbook in AI courses that emphasize programming.

- *Artificial Intelligence and Natural Man* by Margaret Boden. Basic Books (a division of Harper & Row), 1977. $12.95; 537 pages. Paperback.

Written in England nearly a decade ago, this "classic work" on artificial intelligence describes how computers work and details their many uses. Like many of the other books, it explores cognition, reasoning, and other human attributes that relate to artificial-intelligence endeavors. The author is a professor of philosophy and psychology at the University of Sussex. Her book has extensive annotations and bibliographic data.

- *The Binary Brain: Artificial Intelligence in the Age of Electronics* by David Ritchie. Little, Brown and Company, Boston, 1984. $14.95; 212 pages. Hardcover.

Ritchie explores the evolution of the computer and how the human mind works. According to the author, the book grew out of a suggestion from the publisher: "to see a book explore the ideas that computers are the next step in human evolution—a leap forward equivalent to the development of the cerebral cortex." For readers who wish to explore AI and cognition further, the book contains a selected bibliography and a recommended-reading list.

- *Brains, Behavior, Robotics* by James Albus. Byte Books (a subsidiary of McGraw-Hill), 1981. $21.95; 352 pages. Hardcover.

Albus is from the "brain as a highly developed computer" school. According to Albus, "the best robot-control system is one that is modeled after the human brain," and such modeling is difficult. The book deals with potential applications of robot technology—both current and future. Albus uses and lists numerous reference works.

- *The First Artificial Intelligence Coloring Book* by Harold Cohen, Becky Cohen, and Penny Nii. William Kaufmann, Inc., Los Altos, CA, 1984. $19.95; 127 pages. Hardcover.

The crude but intriguing drawings in this book were produced by Harold Cohen and his computer programs. Perforated pages allow you to remove them for framing after you have colored in the abstract images.

- *Robot Vision: International Trends in Manufacturing Technology*, edited by Alan Pugh. IFS Publications Ltd. UK and Springer Verlag, Berlin, 1983. $49.50; 356 pages. Hardcover.

Editor Pugh has pulled together extremely technical material about robotics from various industrial and academic sources. A prefatory note cautions: "This series is intended for manufacturing managers, production engineers and those working on research into advanced manufacturing methods."

- *Industrial Robotics Handbook* by V. Daniel Hunt. Industrial Press, Inc., New York, 1983. $32.50; 432 pages. Hardcover.

Originally published in 1983, this books is in its third printing. It claims to be "the most comprehensive and up-to-date overview available on industrial robots." It's also not the only robotics book from this publisher. Two others: *Robotics Sourcebook and Dictionary* by

David Tver and Roger Bolz, 1983, 330 pages; *Computer Applications in Manufacturing* by Thomas Gunn, 1981, 204 pages.

- *Building Expert Systems*, edited by Frederick Hayes-Roth, Donald W. Waterman, and Douglas B. Lenat. Addison-Wesley, Reading, MA, 1983. $32.95; 444 pages; Hardcover.

Thirty-eight expert-system researchers collaborated to produce the contents of this "broad introduction to the concepts and methods necessary for an understanding of how these systems work." The book is billed as a textbook for computer-science and decision-support students or as a survey for scientists, engineers, and technical managers.

The book's jacket lists ten other AI titles from the publisher, including *Language as a Cognitive Process*, *LISP*, and *Readings in Medical Artificial Intelligence.*

- *Fundamentals of Robotics* by Larry Heath. Reston (a Prentice-Hall Company), Reston, VA, 1985. $32.95; 412 pages. Hardcover.

Textbook publisher Reston has a robotics series, and this book is one of six titles. Heath stresses theory and applications. Other books concentrate on programming, manufacturing, and other aspects of robotics.

What's interesting about this random sampling of AI/robotics books is that quite a few of them are published in England and that a relatively large percentage of them are not strictly textbooks or reference books. (The proximity of the bookstore to Stanford University may have accounted for the display.)

This display, which constitutes only of a portion of the AI/robotics-books universe, indicates that enough books exist to keep even the most avid AI aficionado busy.

Letters

Continued from p. 14.

Debate Continues

Per your reader's complaints concerning your coverage of MS-DOS and PC-DOS, you have to understand: the purchasers of CP/M equipment were perfectly happy with the standard, then along comes IBM, and instead of offering a vastly improved system, they simply offer an incompatible proprietary one.

Then, your magazine: just when CP/M users think they've got something special of their own, Big Blue's DOS rears its ugly head in the only magazine devoted exclusively to CP/M users. I have worked in a publisher's marketing department, so I understand precisely why you must do this, but it hurts nevertheless...

Just promise us that when half the editorial becomes MS/PC-DOS, you'll have the decency to split the books.

Hugh Stone III
Canoga Park, CA

Can't Keep a Secret

I have found your book *Free Software* by sheer luck in one of the local bookstores. I thought that I was a veteran of using free software. I have been downloading all sorts of crazy programs for the last two years. Then I read your book. I could not put it down. It has opened my eyes much wider than before.

Don't you advertise your books? I really *urge* you to advertise (or even announce in your magazine) your books. I regard such advertisements as a public service! Readers of your magazine will be delighted to read these books.

My deepest admiration for writing strong and simple English prose in dealing with computer usage!

Paul Naitoh
Applied Psychophysiology Group
San Diego, CA

The book Free Software *(Bove, Rhodes & Smith, BAEN Books, distributed by Simon & Schuster Mass Merchandise Sales) retails for under $10. Ask for it at your local bookstore — if the store doesn't have it, please tell them to order it.*

User-supported Software Supports Kaypro

I have developed several home application programs for the Kaypro computer. They include KAYPRO ADDRESS BOOK, KAYPRO PHONE BOOK, KAYPRO MEMO MINDER, KAYPRO DIARY, KAYPRO TRIVIA, and KAYPRO HOME OFFICE ORGANIZER. I actually use all these programs and am confident of their quality and usefulness.

I have a special offer that I would like other Kaypro users to know about: send me your name, address, software choice, and which type of Kaypro you use, and I will send you the program. If you like the program, consider its quality and my time, and send me what you think it is worth. If you can't use the program, simply discard the disk or reformat it. All it costs you is the price of a stamp ($.22). All I ask is that since my programs are copyrighted, you do not distribute them. If your friend likes them, have them send me their address etc. and I will extend the same offer to them.

Honest. No strings attached. Of course my main objective is to get *my* Kaypro to pay for itself, but the price of my programs is whatever you think that they are worth.

Joe Yateman
1714 Springfield Drive
Valdosta, GA 31602

MICRO USER'S MARKET

Software/Languages

BDS C - For Developers

Our C compiler was designed especially for the impatient programmer: the compilation/linkage speed leaves every other C compiler in the dust. Execution is really quick, too! BDS C comes with a symbolic debugger, library sources, overlay linker, a MODEM program, and other utilities... 480K of stuff for $150.

BDSoftware, Inc.

BD Software, Inc.
P.O. Box 2368
Cambridge, MA 02238
(617) 576-3828

MBASIC Cross-Reference Utility

BASICREF.COM cross-references variables and line numbers to help you track down redundant or extra variables. FAST!!! Screen, file or printer output. Customizable external dictionary. A must for all BASIC programmers. Just $19.95 (foreign add $5). Most CP/M 5.25" and IBM PC and compatibles.

LTA Associates
7 Ledge Street
Stoneham, MA 02180

Software

Supplement your child's education

with disks prepared by educators for pre-school through middle school. Each offers at least 6 different programs for verbal and non-verbal areas. $27/disk. Programs (MBASIC) run on machines reading Kaypro or Osborne disks. Also for IBM PC, PC Jr. (DOS). We encourage inquiries for descriptive literature.

Maji-Miti
P.O. Box 3244 WVS
Kamuela, HI 96743
SMAIL STE413
(808) 883-9055

User's Guide magazine is introducing a new section of classified ads called **Micro User's Market**, designed to provide affordable advertising. For $245, we will run your 300-character descriptive ad for three consecutive issues; or for $445 we will run your ad for six consecutive issues. Call Annie Warren at (415) 364-0108.

Backup/Restore

Flexible Hard Disk Protection

BackRest provides turnkey backup and restore with full error handling, splitting large files to multiple floppies, bad sector lockout, fault tolerant restoration, informative reports and more. BackRest installs with an interactive installation program. Full DOS Subdirectory and CP/M User support. $180.00.

SSI Stok Software Inc.

Stok Software, Inc.
17 West 17th St.
New York, NY 10011
(212) 243-1444

Books

CP/M User's Guide

Get the complete tutorial on CP/M, CP/M Plus and CP/M-86: *CP/M User's Guide* by Bove & Rhodes for the Jerry Pournelle user's guide series (Baen Books, distributed by Simon & Schuster Mass Merchandise Sales Co.). Order it from your local book store or write to:

Simon & Schuster
Mass Merchandise Sales Co.
1230 Ave. of the Americas
New York, NY 10020

ADVERTISER INDEX

Service	Page
1. Baen Books	7
2. CDE Software	57
3. Compusophic Systems	9
4. Echelon, Inc.	53
5. FOG (First Osborne Group)	23
6. Kaypro Corp.	C2
7. Kaypro User's Group	21
8. Morrow Owner's Review	56
9. New Star	65
10. Orange Microware	9
11. Osborne Computer Corp.	1
12. Paradigm Consultants	37
13. PeopleTalk Associates	15
14. Personal Micro Computers	21
15. Plu Perfect Systems	55
16. Portable Companion	C3
17. SLR Systems	13
18. User's Guide (back issues)	70,71
19. User's Guide (subscriptions)	22
20. Viasyn (formerly CompuPro)	C4

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100

Please complete:
Are you a subscriber? ☐ Yes ☐ No

A. Your computer experience?
1.☐ Novice 3.☐ Intermediate-advanced
2.☐ Semi-intermediate 4.☐ Advanced

B. Occupation?
1.☐ Executive/manager 3.☐ Professional
2.☐ Small bus. owner 4. ☐ Other

C. Use which computer(s)?
a. ☐ Apple b. ☐ IBM PC
c. ☐ CompuPro d. ☐ DEC
e. ☐ Osborne f. ☐ S-100
g. ☐ Radio Shack h. ☐ Heath/Zenith
i. ☐ TeleVideo j. ☐ Sanyo
k. ☐ Kaypro l. ☐ Compaq
m. ☐ Morrow n. ☐ Columbia
o. ☐ ______________ p. ☐ PC-compat.

D. Interested in which software?
a. ☐ Word processing b. ☐ Accounting
c. ☐ Communications d. ☐ Data base mgmt.
e. ☐ Medical/Dental f. ☐ Financial planning
g. ☐ Other vertical h. ☐ Scientific

E. Interested in which products?
a. ☐ Printers b. ☐ Modems
c. ☐ Hard disks d. ☐ Publications
e. ☐ Information services f. Other: ____________

F. Where do you use your computer?
1. ☐ Small business 5. ☐ Medium business
2. ☐ Profession 6. ☐ Home
3. ☐ School/training 7. ☐ Corporation
4. ☐ Construction 8. Other: ____________

G. Your immediate need?
1. ☐ Adding peripherals to present system
2. ☐ Upgrading system performance
3. ☐ Using information/communications services
4. ☐ Supplies and accessories

Name ______________________________ **Age** ______ **Sex** ____
Company ______________________ **Title** ______________________
Address ______________________ **City, State, Zip** ______________________

UG15

FREE INFORMATION

User's Guide Reader Service Card

Each advertisement and product announcement in this issue has a reader service number. To receive mail from the manufacturer, simply circle the same number on this reader service card, and drop the card in the mail. Your name and address is sent on to the appropriate manufacturer, who sends the information.

The service helps you get detailed information about products you may want to purchase. *User's Guide* does not sell or recommend products associated with this service.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100

Please complete:
Are you a subscriber? ☐ Yes ☐ No

A. Your computer experience?
1.☐ Novice 3.☐ Intermediate-advanced
2.☐ Semi-intermediate 4.☐ Advanced

B. Occupation?
1.☐ Executive/manager 3.☐ Professional
2.☐ Small bus. owner 4. ☐ Other

C. Use which computer(s)?
a. ☐ Apple b. ☐ IBM PC
c. ☐ CompuPro d. ☐ DEC
e. ☐ Osborne f. ☐ S-100
g. ☐ Radio Shack h. ☐ Heath/Zenith
i. ☐ TeleVideo j. ☐ Sanyo
k. ☐ Kaypro l. ☐ Compaq
m. ☐ Morrow n. ☐ Columbia
o. ☐ ______________ p. ☐ PC-compat.

D. Interested in which software?
a. ☐ Word processing b. ☐ Accounting
c. ☐ Communications d. ☐ Data base mgmt.
e. ☐ Medical/Dental f. ☐ Financial planning
g. ☐ Other vertical h. ☐ Scientific

E. Interested in which products?
a. ☐ Printers b. ☐ Modems
c. ☐ Hard disks d. ☐ Publications
e. ☐ Information services f. Other: ____________

F. Where do you use your computer?
1. ☐ Small business 5. ☐ Medium business
2. ☐ Profession 6. ☐ Home
3. ☐ School/training 7. ☐ Corporation
4. ☐ Construction 8. Other: ____________

G. Your immediate need?
1. ☐ Adding peripherals to present system
2. ☐ Upgrading system performance
3. ☐ Using information/communications services
4. ☐ Supplies and accessories

Name ______________________________ **Age** ______ **Sex** ____
Company ______________________ **Title** ______________________
Address ______________________ **City, State, Zip** ______________________

UG15

Indicate Offer and Length of your ***User's Guide*** subscription:

Take Advantage of the Low Price for Enclosing Payment Now!	Or, try our no-risk bill-me offer and cancel if not satisfied!
☐ I enclose **$21** for 6 issues	☐ Bill me **$25** for 6 issues
☐ I enclose **$40** for 12 issues	☐ Bill me **$45** for 12 issues

☐ **Foreign:** (prepaid US $$): Canada & Mexico $27/six issues, overseas $40/six issues (airmail).

Payment by charge card: ☐ Visa ☐ Mastercard Exp. date ____________

Card # ____________ Interbank # ____________

Signature required: ____________

Name ____________

Company ____________ Title ____________

Address ____________

City, State, Zip ____________

☐ Check if you do not want promotional mailings.

Make checks payable to ***User's Guide***. Offer expires 12/31/85
Subscription Dept.: P.O. Box 5245, Redwood City, CA 94063.
Please allow 6-8 weeks for processing.

UG15c

Indicate Offer and Length of your ***User's Guide*** subscription:

Take Advantage of the Low Price for Enclosing Payment Now!	Or, try our no-risk bill-me offer and cancel if not satisfied!
☐ I enclose **$21** for 6 issues	☐ Bill me **$25** for 6 issues
☐ I enclose **$40** for 12 issues	☐ Bill me **$45** for 12 issues

☐ **Foreign:** (prepaid US $$): Canada & Mexico $27/six issues, overseas $40/six issues (airmail).

Payment by charge card: ☐ Visa ☐ Mastercard Exp. date ____________

Card # ____________ Interbank # ____________

Signature required: ____________

Name ____________

Company ____________ Title ____________

Address ____________

City, State, Zip ____________

☐ Check if you do not want promotional mailings.

Make checks payable to ***User's Guide***. Offer expires 12/31/85
Subscription Dept.: P.O. Box 5245, Redwood City, CA 94063.
Please allow 6-8 weeks for processing.

UG15b

Indicate Offer and Length of your ***User's Guide*** subscription:

Take Advantage of the Low Price for Enclosing Payment Now!	Or, try our no-risk bill-me offer and cancel if not satisfied!
☐ I enclose **$21** for 6 issues	☐ Bill me **$25** for 6 issues
☐ I enclose **$40** for 12 issues	☐ Bill me **$45** for 12 issues

☐ **Foreign:** (prepaid US $$): Canada & Mexico $27/six issues, overseas $40/six issues (airmail).

Payment by charge card: ☐ Visa ☐ Mastercard Exp. date ____________

Card # ____________ Interbank # ____________

Signature required: ____________

Name ____________

Company ____________ Title ____________

Address ____________

City, State, Zip ____________

☐ Check if you do not want promotional mailings.

Make checks payable to ***User's Guide***. Offer expires 12/31/85
Subscription Dept.: P.O. Box 5245, Redwood City, CA 94063.
Please allow 6-8 weeks for processing.

UG15a

Subscribe Now

No Postage
Necessary
if mailed
in the U.S.

BUSINESS REPLY CARD
FIRST CLASS PERMIT #36 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

User's Guide BACK ISSUES

PO Box 5245
Redwood City, CA 94063

Billing Address:

No Postage
Necessary
if mailed
in the U.S.

BUSINESS REPLY CARD

FIRST CLASS PERMIT #36 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

User's Guide
P.O. Box 5245
Redwood City, CA 94063

Act now to complete your set of ***User's Guide*** back issues! Limited offer while supplies last! This may be your last chance to get a complete set!

BACK ISSUES BACK ISSUES BACK ISSUES

Order now while supplies last! Payment must accompany order:

Issues #1-#4	Qty.	at **$6** each =	$
Issues #5-#10	Qty.	at **$5** each =	$
Issue #11/12	Qty.	at **$8** each =	$
Issues #13-#14	Qty.	at **$4.50** each =	$
First Class Postage	Qty.	at **$1.50** each =	$
Foreign airmail add	**$3** per copy or **$10** per 4 copies =		$
		TOTAL ORDER:$	

Payment by charge card: ☐ Visa ☐ Mastercard Exp. date

Card # Interbank #

Signature required:

Name

Company Title

Address

City, State, Zip

☐ Check if you do not want promotional mailings.

Make checks payable to ***User's Guide***. Offer expires 12/31/85
Back Issues Dept.: P.O. Box 5245, Redwood City, CA 94063.
Please allow 6-8 weeks for processing.

UG 15b/ia

Billing Address:

No Postage
Necessary
if mailed
in the U.S.

BUSINESS REPLY CARD
FIRST CLASS PERMIT #36 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

User's Guide
P.O. Box 5245
Redwood City, CA 94063

Billing Address:

No Postage
Necessary
if mailed
in the U.S.

BUSINESS REPLY CARD
FIRST CLASS PERMIT #36 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

User's Guide
P.O. Box 5245
Redwood City, CA 94063

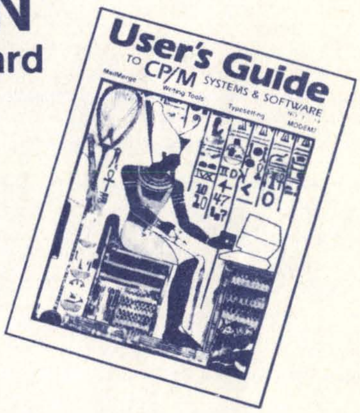
No Postage
Necessary
if mailed
in the U.S.

BUSINESS REPLY CARD
FIRST CLASS PERMIT #36 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

User's Guide
PO Box 5245
Redwood City, CA 94063

FREE INFORMATION
User's Guide Reader Service Card



Here is an easy way for you to get manufacturers' information about products advertised or mentioned. Circle the numbers on the card that correspond to key numbers at bottom of advertisements and fill out and mail the postage paid card and the literature will be sent to you free of charge.

No Postage
Necessary
if mailed
in the U.S.

BUSINESS REPLY CARD
FIRST CLASS PERMIT #36 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

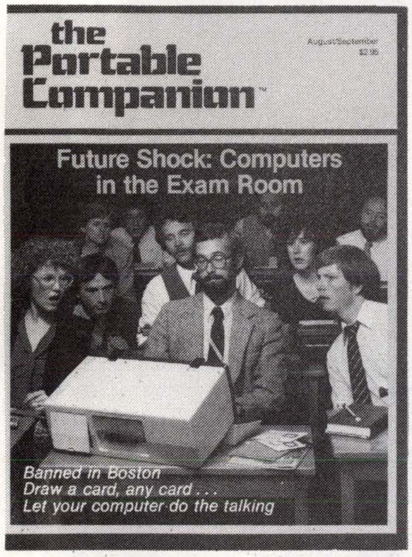
User's Guide
PO Box 5245
Redwood City, CA 94063

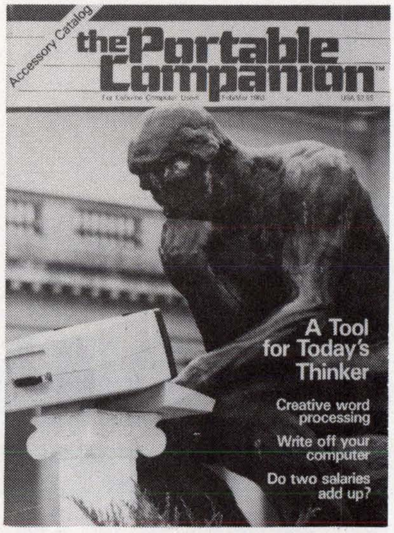
EDUCATE YOUR OSBORNE!

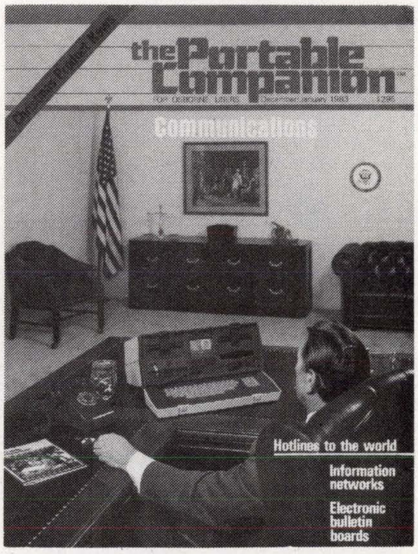
Subscribe Now to *THE PORTABLE* **Companion!**

FOR OSBORNE COMPUTER USERS

Can you teach an Osborne new tricks? You can get the most from your investment in your Osborne computer by subscribing to *THE PORTABLE* **Companion**. This magazine is distributed by subscription only, so order today!


the Portable Companion™
August/September
$2.95
Future Shock: Computers in the Exam Room
Banned in Boston
Draw a card, any card . . .
Let your computer do the talking


Accessory Catalog
the Portable Companion™
A Tool for Today's Thinker
Creative word processing
Write off your computer
Do two salaries add up?


the Portable Companion™
December/January 1983
Communications
Hotlines to the world
Information networks
Electronic bulletin boards

☐ Send me **six issues of The Portable Companion** for $25, payment enclosed,

OR ☐ Bill me $28 for **six issues of The Portable Companion.**

☐ **Foreign** (prepaid US $$): Canada & Mexico $35/six issues, overseas $50/six issues (airmail).

Payment by charge card: ☐ Visa ☐ Mastercard Exp. date ____________

Card # ____________ Interbank # ____________

Signature required: ____________ **TOTAL ORDER:** $____________

Name ____________

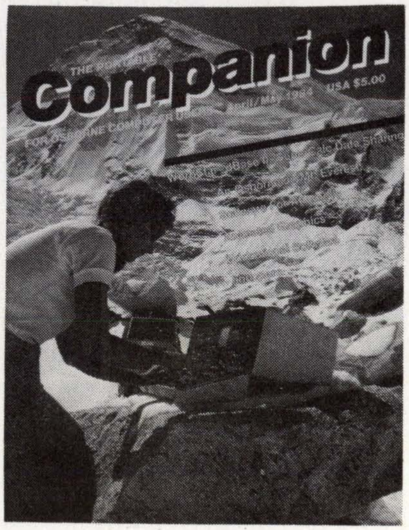
Company ____________ Title ____________

Address ____________

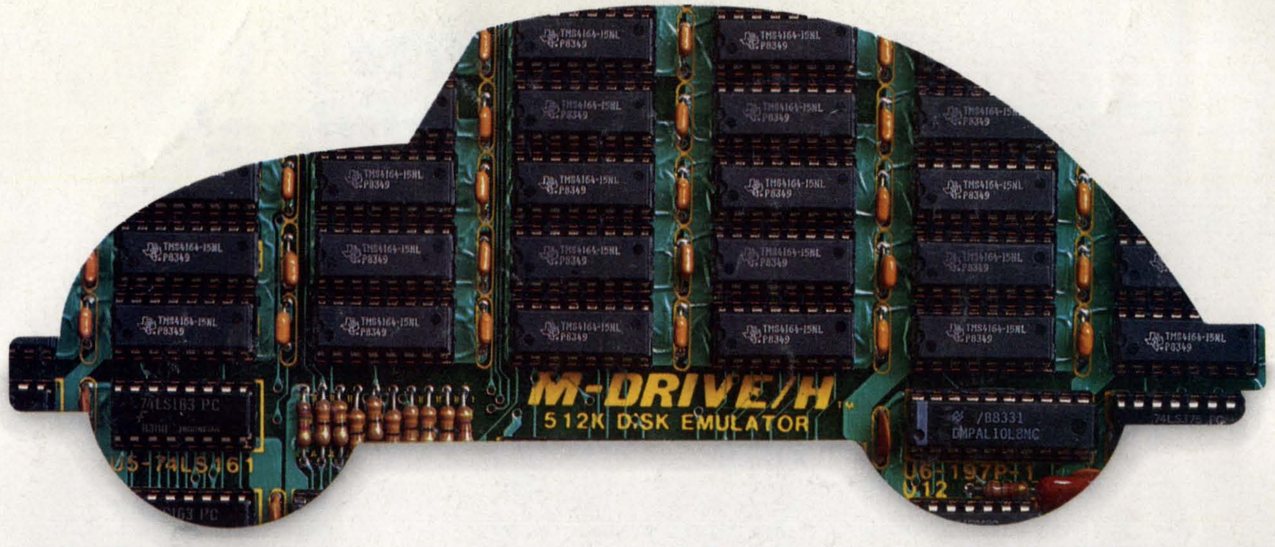
City, State, Zip ____________

☐ Check if you do not want promotional mailings.

Make checks payable to *Portable Companion.* Offer expires 12/31/85
Subscription Dept.: 42680 Christy St., Fremont, CA 94538. PC16d
Please allow 6-8 weeks for processing.


Companion
USA $5.00

Circle number 16 on the reader service card.



You can still buy quality and dependability at a reasonable price.

That's exactly what we drive home at Viasyn. We offer you a whole line of CompuPro® IEEE 696/S-100 Bus boards. Along with single- and multi-user systems compatible with over 3,000 standard business applications, plus a wide variety of scientific and industrial programs. Each one's a value our competition finds hard to beat.

You can choose from a broad range of CPU boards, memory boards, disk controllers, network and interface boards, plus PC Video, in color or black and white. You can mix or match 8-bit and 16-bit software on the same machine or on different machines. And pick just the power and memory you need. Object: to grow and multiply in capabilities, at optimum cost efficiencies. Without sacrificing quality or dependability.

If that's what you're driving for, write us for our new short-form catalog or simply call our toll-free number.

VIASYN™
The CompuPro People
Where Computers Grow

3506 Breakwater Court, Hayward, CA 94545
Call 800/VIASYN-1. In CA, 800/VIASYN-2. TWX: 510-100-3288 VIASYN CORP

CompuPro is a registered trademark of Viasyn Corporation.

Circle number 20 on the reader service card.